D3

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 May 2001 (03.05.2001)

PCT

(10) International Publication Number
WO 01/31492 A2

(51) International Patent Classification[7]: G06F 17/00

(21) International Application Number: PCT/US00/41361

(22) International Filing Date: 20 October 2000 (20.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/429,853 28 October 1999 (28.10.1999) US

(71) Applicant: ONFLOW CORPORATION [US/US]; Suite 300, 160 Pine Street, San Francisco, CA 94111 (US).

(72) Inventors: NIFFENEGGER, Bill, E.; One St. Francis Place #6201, San Francisco, CA 94107 (US). THOENNES, Joseph, P.; 85 Rossi Avenue, San Francisco, CA 94118 (US). TUTTLE, Douglas, D.; 170 Pacific Unit 41, San Francisco, CA 94111 (US).

(74) Agents: RAUBVOGEL, Amir, H. et al.; Fenwick & West LLP, Two Palo Alto Square, Palo Alto, CA 94306 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/31492 A2

(54) Title: ONLINE FOCUSED CONTENT GENERATION, DELIVERY AND TRACKING

(57) Abstract: A method, system, and computer program product provide authoring, generation, delivery, authentication, and tracking for online content. Delivered content is personalized based on demographic information describing the user, client information describing the hardware and software environment at the client, and other factors. Content is generated and displayed using a plug-in for a web browser, capable of outputting premium-level content without requiring large amounts of bandwidth or download times. Client views of and interactions with displayed content are monitored and tracked, an may be used as a basis for advertising billings, sales, or other commercial interactions.

Online Focused Content Generation, Delivery and Tracking

Field of the Invention

The present invention is related to online content, and more particularly to a method, system, and computer program product for providing authoring, generation, delivery, authentication, and tracking for such content.

Description of the Background Art

The World Wide Web is increasingly becoming an advertising-driven medium. With the growing commercialization of the Web, web pages containing banner ads and pop-up ads have become ubiquitous. Typically, such ads will be displayed on a user's screen alongside or on top of other content, so as to force the user to see the ad when viewing the other content. In many cases, web pages are constructed so that ads become visible several seconds before the remaining content appears, so as to further increase the amount of time a user spends viewing an ad.

In general, advertisers are interested in tracking user behavior concerning their ads, and are particularly interested in determining which demographic groups may be interested in their products or services. By targeting specific demographic groups and monitoring the results, advertisers can increase the effectiveness of their advertising and of its placement, and can customize ads so as to appeal to particular audiences. Thus, advertisers expend considerable effort to obtain information regarding user's interaction with their ads.

For online advertising as found on the World Wide Web, advertisers can track certain information concerning user behavior and ad penetration, using techniques that are known in the art. For example, a content provider may provide an advertiser with a "hit count" indicating how many "GET" requests for the ad were received in a given time period. However, such "hit counts" do not provide accurate information regarding whether the ad was actually viewed by each user; for example, the ad may not have been received by the user's browser, the user may have turned the graphics

off on his or her browser, the user may have scrolled to another portion of the page, or he or she may have ignored the ad altogether.

As is also known in the art, content providers may provide advertisers with information as to the number of "clickthroughs" associated with an ad (i.e. the number of times a user clicked on the ad to be taken to another page). Such information provides a rudimentary level of tracking user behavior, but only captures a small fraction of interested users. Many users who view an ad do not click on the ad, whereas other users may tend to click on everything they see, so that their clickthroughs may not be as significant as others. Furthermore, the demographics of the particular users who have clicked on the ad are generally not known.

More sophisticated attempts at monitoring user behavior are also known in the art. Many websites employ "cookies" (small files stored on the user's local hard drive) to identify users and to track repeat visitors to the site. If information is known about a user based on previous visits to the site, then the user's experience can be customized so as to be more likely to appeal to his or her interests. On some sites, such as for example **my.yahoo.com**, users fill out onscreen forms to indicate what type of information interests them. A cookie is stored on the user's hard drive so as to identify the user for future visits. The website can then provide content (and advertising) according to the previously-provided user information.

Though useful, cookies are of limited value because they only operate within a given domain. A cookie stored by a particular website can generally only be accessed by that website or by a related site, and presumably contains information that can only be interpreted by an application running on that website's server (for example, the cookie may store a user ID which identifies the user in a proprietary database). Thus, cookies do not provide a mechanism by which advertisers can track user behavior with respect to their ads over a number of diverse websites.

In general, conventional browsers have no generalized mechanism for reporting user access to content, including views, interactions, how long the content was visible, and the like.

Furthermore, conventional browsers are in general incapable of presenting high-quality graphics, sound and interactivity. In some cases, graphics and animations may be presented, but only if substantial bandwidth is available for download-

ing large files. Plug-ins may be available for displaying special effects, but none of the existing systems are able to present content at a premium level using minimal bandwidth.

In addition, as the popularity of the Internet continues to grow, new hardware devices and environments are becoming available for accessing the Web. In many cases, the particular characteristics, capabilities, and limitations of these devices and environments can vary widely. For example, a web-enabled cell phone may provide an entirely different browsing experience than would a desktop computer running a state-of-the-art browser. Connection speed and bandwidth may also affect the user's browsing experience. No generalized scheme currently exists for providing truly customizable content based on the capabilities and limitations of the user's environment.

What is needed is a system, method, and computer program product for generating personalized content (such as ads) for users, based on user characteristics such as demographics, psychographics, and the like, as well as on hardware, software, and connection characteristics.

What is further needed is a system, method, and computer program product for tracking user behavior with respect to delivered content, in a manner which is domain-independent and which provides greater flexibility and improved information.

What is further needed is a system, method, and computer program product for displaying premium-level content, including graphics, sound, and interactivity, without requiring large amounts of bandwidth and/or download time.

What is further needed is a system, method, and computer program product for providing an authoring environment facilitating efficient and easy authoring of sophisticated and complex content.

Summary of the Invention

The present invention provides a multimedia authoring and playback system which avoids the limitations of the prior art and provides additional flexibility with regard to content generation and tracking. A player client is built into the user's browser (or provided as a plug-in), allowing presentation of complex and interactive content from within the browser window. When a user accesses a page containing

content to be played by the player client, the player client communicates with a content server to obtain the content. Content may be provided to the player client in the form of instructions in a scripting language. By moving some of the processing load for rendering content from the server to the client, the present invention allows sophisticated content to be presented without requiring downloads of large quantities of data.

The player client monitors user behavior with respect to the content it displays. Information describing such behavior is transmitted to a data collection server for tracking purposes. Each player client is given a unique identifier so that the data collection server can track behavior more effectively. At initial installation of the player client, or at some later date, descriptive information about the user is collected, so that the player client identifier can be associated with such information. Thus, advertisers can be given detailed information as to the demographics and behaviors of the users viewing their ad content. Complete demographic reporting and data mining are provided.

The player is also able to communicate information describing whether an ad is visible on the screen, how long the ad is visible, and what user interaction, if any, has taken place with respect to an ad. Based on the information provided by the player to the data collection server, advertisers can customize content, ad rates can be determined (depending on the quantity and demographics of users viewing the ad), and profiles of users can be developed. Ads and other content are served from a central server, and the publisher of an ad is charged based on monitored access to the ad content, as well as other user behavior such as click-throughs and other interactions. Advertisers can be charged on a per-transaction basis, with details of each user interaction with the ad being used to determine rates.

The present invention thus encompasses any or all of the following components: a player client for providing data, graphics, sound, interactivity, tracking, and authoring at the client end; a content server for serving content in response to requests from the player client; an author server implemented as a hosted application for creating, modifying, and publishing content; a data collection server for receiving and tracking content "serves" and user behavior with respect to served content; and a knowledge base for analyzing data collected from player clients and other sources.

The present invention is capable of displaying premium-level content without requiring large amounts of bandwidth or download time. In one embodiment, for example, the present invention can display broadcast-television-quality text and effects, including lighting, warping, glows, blurs, 2 1/2 D rotations, transitions, and the like. Additional effects include sophisticated interactivity, anti-aliased and full transparency, and high frame rates for smooth animations. User input and interaction can be monitored and tracked to any desired degree of detail. The present invention is extremely flexible and easy to use, and employs an object-oriented data flow architecture that provides a high degree of modularity for maximum flexibility and power in authoring capability.

The present invention also provides a system for verifying the integrity of served content by performing an authentication check before playing the content. If the content fails the check, it is not played, thus preventing illicit content modifications (as well as protecting against transmission errors).

The present invention provides functionality for identifying each of a number of users of a particular client machine and for personalizing content based on information collected about each user. Any number of "personas" can be identified and associated with a client machine, so that the content server can provide content appropriate to the currently active persona. This increases the likelihood that the end user will be interested in the content, and thereby improves advertising targeting, sales, consumer interest, and the like.

The present invention further provides functionality for identifying particular types of client machines and associated browsers. Thus, content can be personalized based on the particular characteristics and limitations of the client hardware and software. For example, content that is displayed on devices such as cell phones or wireless PDA's could be presented in a format that is specifically tailored to the small screen sizes and limited resolution of such devices. By personalizing content based on the user machine's limitations, the present invention provides a technique for maximizing the quality of the user's online experience.

The present invention further provides functionality for authenticating delivered content and for preventing unauthorized content from being displayed or played.

The present invention provides an authoring environment for generating content in an intuitive, efficient, and easy-to-use manner. Content is represented internally using a data flow graph, and constructed using a polymorphic object-oriented software architecture for maximum flexibility and robustness.

Brief Description of the Drawings

Fig. 1 is a block diagram of a system architecture according to one embodiment of the present invention.

Fig. 2 is a flowchart of a transaction including a content request, generation, and display according to one embodiment of the present invention.

Fig. 3 is a block diagram of a set of resources for practicing the present invention.

Fig. 4 is a block diagram of an authoring operation according to one embodiment of the present invention.

Fig. 5 is a block diagram of a publishing operation according to one embodiment of the present invention.

Fig. 6 is a block diagram of a content playing operation according to one embodiment of the present invention.

Fig. 7 is a block diagram of a data transfer operation according to one embodiment of the present invention.

Fig. 8 is a block diagram showing a site structure for downloading an authoring environment according to one embodiment of the present invention.

Fig. 9 is a diagram of a screen containing an authoring environment provided by the present invention.

Fig. 10 is an example of content that can be customized by an Author and branded or sponsored before publication.

Fig. 11 is an example of a customization operation for content.

Fig. 12 is an example of a user's project screen.

Fig. 13 is an example of a sheet containing solutions for themed projects.

Fig. 14 is an example of a control palette for a content element.

Fig. 15 is an example of hint help for a control palette.

Fig. 16 is an example of a nonperiodic waveform.

Fig. 17 is an example of a periodic waveform.

Fig. 18 is an example of a content element implemented using a data flow graph.

Detailed Description of the Preferred Embodiments

The following description presents the system of the invention in the context of advertising that is served over a network and displayed on web pages. However, one skilled in the art will recognize that any type of content, whether or not related to advertising, and whether or not displayed on a web page, can be personalized, delivered, and tracked in a manner similar to that described below, without departing from the spirit or essential characteristics of the present invention.

Definitions

Unless otherwise defined herein, all terms which are commonly used in the computer, marketing, and Internet communities shall have the meanings commonly given such terms in such communities. The following definitions are provided for illustrative purposes only, and are not intended to limit the scope of the invention as claimed herein.

Project – An editable file (extension .ofb) including source objects and internal user interface (UI) controls.

Source Objects – The set of objects used to construct a specific Project. This can include text files, images, scripts, proprietary-format elements, sounds, and the like.

Element - A tool, graphic, effect, or process embedded in content files. An element can contain parameter controls used during authoring. Elements are selected by an Author from sheets, as described below in connection with Fig. 13.

Internal User Interface (UI) Controls – Interactive graphical elements embedded in content files. These are used, for example, by Authors to manipulate effects by varying parameters.

Role – A permission descriptor for system objects. Roles such as Author, Manager, or Publisher are assigned to a Project and each role has a specific characteristic set of permissions with respect to the Project.

Originator – The Author who creates a new Project file. Initially, the Originator is also the Manager and sole Author of the Project. If the Originator is a registered Publisher, he or she is also the sole Publisher of the Project.

Manager – A role including all permissions for the Project. In one embodiment, there is only one Manager per Project. Initially, the Manager is the Originator. The Manager can assign any Project permissions including Management, Authoring, Publishing, and Viewing.

Author – A role including editing rights for a Project. There can be multiple Authors per Project. An Author is given access to a Project when he or she creates a new Project, or when the Manager of an existing Project assigns access to the Author. An Author can also be a Manager of a specific Project by 1) creating a new Project or 2) having the Manager of an existing Project assign Management of the Project to the Author.

Publisher – A role including rights to publish a Project to the web (i.e., to create Publications and Articles). Publisher rights are assigned to a Project by the Manager.

Reviewer – A role including only viewing rights on a project.

Publication – A content entity, or Publishing space, that may contain any number of Articles. A Publication is identified by a PUB number (described below).

Article – A published content file (extension .ofb). An Article is identified by a unique embedded ART ID (described below). An Article is a specific Instance of a project with source files and internal UI controls removed; in other words, it is the end result that is delivered to a player client.

Demographic – A target group to which a publication can be directed.

Instance – A specific delivery, or "serve", of an Article to a player. An Instance is identified by its unique TID (described below).

Publishing – Creating a Publication containing one or more Articles derived from Projects. Creates an HTML <embed> tag that can reference an Article or Articles.

Loading – Sending a Publication to the content server, creating appropriate file links, verifying load, and delivering correct publication Uniform Resource Identifier (URI) to publisher for inclusion in host HTML web pages. The load operation proc-

esses the publication job and configures the server environment for serving the content.

Serving Content and Tracking

Referring now to Fig. 1, there is shown a block diagram of a system architecture according to one embodiment of the present invention. Player client 105 is a plug-in for browser 104, which may be a conventional browser such as Netscape Navigator or Microsoft Internet Explorer. In an alternative embodiment, player client 105 may be provided as a functional component included in (or bundled with) an improved browser. In yet another embodiment, player client 105 may be provided as a stand-alone software product. One skilled in the art will recognize that the player client of the present invention may be provided in the context of any type of software environment, such as for example, e-mail application, screen saver, cell phone or PDA application, terminal, set top box, and the like. Accordingly the following description, which sets forth the details of the invention in the context of a plug-in for a browser, should not be considered limiting of the scope of the present invention.

Player client 105 is capable of interpreting instructions from content server 102 and displaying content via output device 106, which is typically a display screen and/or audio output device. In one embodiment, instructions from content server 102 are provided as a binary content file which is used by player client 105 to construct an internal representation of a data flow graph for performing the desired functionality. The details of the operation of player client 105 will be described below.

Content server 102 provides content, in the form of instructions and/or static files, for display by player client 105. Content server 102 accepts content requests from player client 105 and responds by sending the requested content. In one embodiment, content server 102 sends information to data collection server 103 whenever a content request is made, so that content requests can be tracked in knowledge base 306.

Data collection server 103 collects data describing users and transactions. Such data is used by advertisers and content providers for tracking purposes and to obtain profile information regarding users, computer configurations, and the like.

Knowledge base 306 stores data collected by server 103 and makes such data available for tracking purposes. In one embodiment, knowledge base 306 organizes stored data in the form of transaction tables facilitating tracking, billing, and demographic analysis. Transaction tables may include, for example, detailed information describing users, their hardware and software environments, their patterns of access to content, their interactions with content, and the like. For, example, player client 105 may monitor how long a particular content element was viewed by a particular user, and how many times the user clicked within the area of the content element; such information would be collected by server 103 and stored in knowledge base 306 to be used in billing the advertiser.

In one embodiment, player client 105 monitors user behavior in the following manner. Player client 105 is called by browser 104 to draw each frame of content, such as for example an animation. The user's computer provides player client 105 with numerous pieces of information regarding user behavior, including for example mouse motion, mouse clicks, keystrokes, mouse overs (the amount of time the mouse spends over the window belonging to player client 105, including the exact times the mouse enters and exits the window), and the like. Player client 105 obtains this information by querying the user's computer. Other obtained information includes, for example, the focus state of the window associated with player client 105 (the top window is said to have "focus"), the scroll position of the window, the URL of the HTML page that contains the <embed> tag that initiated the content, and the like.

Web server 107 is a conventional server (such as the Microsoft Internet Information Server, or the Apache server) for providing pages 108 on the World Wide Web. Browser 104 requests pages 108 from web server 107 and receives requested pages for display on output device 106, as is known in the art.

Servers 102 and 103 are implemented using server software and hardware as is known in the art, such as for example Sunsparc, Solaris, Apache web server, and the like.

In one embodiment, player client 105 is installed when a user downloads and activates an installation program. At the time of installation, or at some later date, player client 105 may prompt the user for personal information which may be useful to advertisers. Such information is transmitted, in one embodiment, to data collection

server 103, along with a unique identifier for player client 105. Thus, a user profile can be constructed and uniquely associated with the identifier for player client 105. In one embodiment, if the user refuses to provide such information, player client 105 can periodically re-query the user at predetermined intervals.

As is known in the art, the user employs browser 104 to access web pages 108 from web servers 107. Web page 108 is delivered to browser 104, which then displays the contents of page 108 on output device 106. As is known in the art, various forms of content can be included on web pages 108, such as graphical elements, Java applets, sounds, and the like. To include such content, web page 108 may direct browser 104 to retrieve a file or other data, either from web server 107 or from some other source.

For example, if an image is to be included on web page 108, the Uniform Resource Locator (URL) for the file containing the image is included in the source code for page 108, so that browser 104 can retrieve the image from the designated source. Information specifying the placement and sizing of the image within page 108 is also provided in the source code. Once the image is retrieved, browser 104 displays page 108, including the image, on output device 106.

In one embodiment of the present invention, certain types of content are associated with and passed on to player client 105 for display. Thus, web page 108 may specify that a certain region 109 contains an ad (or other content) to be displayed using player client 105. In one embodiment, this is accomplished using an HTML "<embed>" (or equivalent) tag specifying the type of content and providing a pointer to the location on content server 102 where the content may be found.

By storing user profile information and transaction data on knowledge base 306, the present invention allows advertisers and content providers to track usage across domains, without being limited to a particular domain as is generally the case with conventional cookies. Profile information can be matched with transaction data, so knowledge base 306 can provide detailed information on who is accessing particular ads and how they are interacting with the ads, as well as providing detailed user information including profiles, demographics, psychographics, and the like. Ad content delivered to users can be targeted and focused based on this detailed information as well as previous behavior.

In addition, since knowledge base 306 contains a log of all transactions with respect to a particular ad, advertising rates can be based on the information derived from knowledge base 306.

In one embodiment, knowledge base 306 further stores information describing the configuration of player client 105 and of the user's hardware environment. Storing such information allows advertisers to obtain detailed information regarding user hardware environments, so that content can be tailored to particular environments, if desired.

Referring now to Fig. 2, there is shown a flowchart of a transaction including a content request, generation, and display according to the present invention. One skilled in the art will recognize that the steps described in Fig. 2 are merely exemplary, and that other combinations of steps and operations may be possible without departing from the spirit or essential characteristics of the present invention.

First, browser 104 requests 201 a web page from web server 107 in a conventional manner, as is common for web-based transactions (e.g., a part of HTTP protocol "GET" request). In response to the request, web server 107 provides 202 the requested page (e.g. an HTML file) by transmitting it to browser 104. Browser 104 receives 203 the page.

When browser 104 detects 204 an <embed> (or equivalent) tag specifying content that is to be played by player client 105, it activates 205 player client 105, passing it the information contained in the <embed> tag. This information includes, for example, an identifying code (or target location) specifying the content, as will be described in more detail below.

<embed> tags provide a mechanism for "plug-ins" to display content within a browser window. When a browser detects an <embed> tag, it activates a plug-in identified by the "type" value within the <embed> tag. The browser then requests (from a web server) any additional content as specified by the "src" value within the <embed> tag. This may include, for example, sound files, animations, applets, and the like.

In one embodiment of the present invention, browser 104 reads the <embed> tag and requests the appropriate content as specified by the "src" value, as is known in the art. In another embodiment, player client 105 obtains 206 the target location

from the <embed> tag, and requests 207 the content from content server 102. In making this request, player client 105 identifies itself using a user ID or player ID, so that the content may be personalized. For example, if the user ID is associated with a teenager, content server 102 may provide more youth-oriented content than if the user ID is associated with a 35-year-old.

In one embodiment, this is accomplished by providing an "ofb" value in the <embed> tag which replaces the "src" tag and is meaningless to browser 104, but which can be resolved by player client 105. For example, the <embed> tag may be of the following form:

<embed ofb="?12345" type="video/x-onflow">

The type of "video/x-onflow" specifies that the content is intended for player client 105. The ofb of "?12345" is unresolvable by browser 104, so that browser 104 will make no attempt to retrieve content based on this ofb value. However, player client 105 is capable of resolving the ofb value to a particular URL (such as onflow.com/playercontent/12345.ofb), so that it can make the appropriate request to content server 102. In addition, the use of this format for the ofb value shortens the overall <embed> tag, thus decreasing the likelihood of a typographical error being introduced, and reducing the amount of data passed between web server 107 and browser 104.

In order to allow content server 102 to personalize the content sent to player client 105 based on various factors such as demographic, psychographic, and other characteristics of the user, historical information, client hardware and software characteristics, and the like, player client 105 may, in one embodiment; embed identifying information in its request to content server 102. For example, for tag <embed ofb="?12345" type="video/x-onflow>, player client 105 generates a request to content server 102 using a URL path such as:

http://www.onflow.com/00/01/12345.ofb

The subdirectory "00/01" is selected based on a persona or demographic ID (or psychographic ID, or some other desired subset) associated with player client 105. In one embodiment, each player client 105 has a demographic ID that is used whenever any request is made to content server 102. In another embodiment, the demographic ID associated with a particular player client 105 may change over time or ac-

cording to time of day. For example, player client 105 may use a demographic ID of "00/01" during certain hours when teenagers tend to be using the computer, and may use a demographic ID of "00/02" during other hours when older members of the family tend to be using the computer. In addition, demographic IDs may be associated with different user profiles, so that when a particular user signs on to the computer at startup, the appropriate demographic ID for that user becomes active. In addition, demographic ID or persona can change over time in response to user behavior and based on analysis of tracking information in knowledge base 306. This is implemented, for example, by performing transaction analysis and cross-referencing the results with other data to produce and maintain meaningful demographic IDs. If necessary, new demographic ID information is transmitted to player client 105 when appropriate, so that player client 105 can use the new ID when generating future content requests.

The demographic ID or persona may be stored at the content server 102 for each particular player client 105, or it may be provided by player client 105 when a content request is made.

Thus, depending on the currently-active demographic ID or persona for player client 105, the generated URL path may differ. Content server 102 resolves each of these URL paths to a particular content file to be transmitted back to player client 105. For example, the following structure may be provided:

- //00/01/1234.ofb resolves to //content/1234a.ofb
- //00/02/1234.ofb resolves to //content/1234a.ofb
- //00/03/1234.ofb resolves to //content/1234a.ofb
- //00/04/1234.ofb resolves to //content/1234b.ofb

In one embodiment, a structure such as the above-described example is implemented using pointers in a file system (also known as "hard links"). Such a technique permits content server 102 to respond to requests, and provide personalized content, with a minimum of processing or redirecting. A similar technique may be employed to personalize content based on hardware characteristics of the client machine (such as particular limitations of cell phones or PDA's, for example), psychographic characteristics, and other factors. The use of a multi-level directory structure,

as shown in the above examples, facilitates implementation in a multi-dimensional personalization space.

Alternative mechanisms for personalizing content may also be employed. For example, player client 105 may provide its unique Player ID to content server 102, and content server 102 may then determine which type of content to provide based on demographic and historical information. One skilled in the art will recognize that many techniques are available for performing this personalization.

In one embodiment, content server 102 is duplicated at several "mirror" sites in various locations, so that content can be requested from an appropriate location, considering relative geographical proximity, load balancing, and other factors. Player client 105 can also select a mirror site corresponding to the source of the original player client 105 when it was initially downloaded and installed.

In response to the request from player client 105, and taking into account the requested content as well as user ID demographics, content server 102 provides 208 content by transmitting a set of instructions to player client 105. In one embodiment, such instructions are in a proprietary format for the player client 105 to interpret and execute in order to display content. In another embodiment, such instructions are provided in a conventional file format for display in a conventional manner.

In one embodiment, player client 105 compares the domain of the page on which the <embed> tag is located with a list of authorized domains obtained from the published content. If the domain is not in the list, the content is not played. Thus, player client 105 provides an additional level of authentication to ensure that specialized content is only made available to authorized domains.

Content server 102 also logs 209 the content request. In one embodiment, each "serve" (request for content and associated response) is given a unique transaction ID. Content server 102 provides the transaction ID and associated information describing the content request to data collection server 103. Content server 102 also provides player client 105 with the transaction ID, so that player client 105 can later include this ID in the data sent to data collection server 103. This allows data collection server 103 to track later-received information from player client 105 describing user behavior with respect to the received content. This later-received information can be matched up with the log event for the original "serve" using the associated

transaction ID, for tracking purposes and for storage in knowledge base 306. In addition, the unique transaction ID provides improved security, by facilitating verification that the content came from an approved content server 102.

Since in one embodiment, each "serve" of a given content item is given a unique transaction ID, the content server 102 of the present invention is able to detect caching of content at the browser 104 or player client 105 by the repetition of a given transaction ID. Content server 102 is further able to detect situations in which a given content item is not properly received or displayed by player client 105, since the transaction ID is not reported back to content server 102.

In an alternative embodiment, content "broadcasting" is facilitated, in which content server 102 transmits a given content item to a large number of player clients 105. If desired, the same transaction ID can be used to identify a single broadcast to a large number of clients 105.

Once player client 105 receives 210 the content, it outputs 211 it. In one embodiment, player client 105 is a real-time rendering engine that is capable of detecting its focus status (i.e. whether it is the topmost window), and can further detect whether any portions of its windows are obscured, offscreen, scrolled out of view, or overlapped by other windows. In one embodiment, player client 105 provides active content and animations when the content area is actually visible on the screen, and does not provide active content when the content area is invisible, scrolled off-screen, or partially or wholly obscured. Player client 105 is further able to detect and record user interactions, including mouse-clicks, mouse movements, rollovers, text input, and the like. Authors of content can specify what types of user interactions and/or events are of interest; in response to such specification, player client 105 detects and transmits information concerning the specified interactions.

Player client 105 records 212 such information concerning its focus status and user interaction. After the content has been displayed, player client 105 transmits 213 information describing the transaction, including for example:

- which player played the content (identified by serial number);
- when the content was played (in local time);
- how long the content was visible on the screen (not including time spent scrolled off screen, not in the active window, or behind another window);

- what user interactions may have occurred with respect to the content; and
- any other interactions and/or events specified by the Author of the content.

Such information is transmitted to data collection server 103 and stored in knowledge base 306. In one embodiment, transmission 213 occurs when the user closes the page containing the content. In another embodiment, transmission 213 occurs when the user closes browser 104, so that the information may be aggregated for a number of transactions and transmitted. In the case of an abrupt disconnection or other "dirty" shutdown, or in any other situation where player client 105 does not have the opportunity to transmit the information at the appropriate time, the data is stored locally and transmission is performed the next time browser 104 starts up, or the next time player client 105 requests content from content server 102. In alternative embodiments, the information is transmitted to data collection server 103 at other times, such as at periodic intervals during the browser session.

The information describing user interaction and behavior is transmitted to data collection server 103 using any desired format. In one embodiment, the information is statistically aggregated (e.g. averaged) and converted to strings, for transmission to data collection server 103. The local time of transmission is also included, if desired. In one embodiment, the information is transmitted as part of a GET command, thus enabling transmission through firewalls, and may take a form such as:

GET /Onflow?01.01.0028=1-12345=0-12345=[30/Sep/1999:08:23:35-0700]=CASE_UNIT_ROTATION_ANGLE_m180.html=4.96=0 HTTP/1.0

If desired, the transmission may be encrypted to prevent interception or counterfeiting.

Once data collection server 103 receives information describing a transaction, the transaction is considered complete. The advertiser can be billed, based on the user profile, how long the ad was viewed, what type of interactions may have occurred with respect to the ad, and the like.

Content

In one embodiment, content is provided in files having a particular extension, such as ".ofb". A content file contains all the information required to play a particular piece of content, including for example sounds, movies, animation, and interactivity. In one embodiment, the authoring user interface is included in the content file as well, as described below. The content file is compressed using conventional compression methods, to improve download times.

Content is developed using either a scripting language or an authoring environment, as described in more detail below. The scripting language or authoring environment develops a content file "behind the scenes", so that the Author need not be aware of the details of the implementation. Additional details on the operation of the authoring environment and scripting language are provided below. When delivered to a player client 105, the information in the content file is used to develop an internal representation in machine-understandable format. In one embodiment, this internal representation takes the form of a data flow graph consisting of nodes and connections among nodes. Player client 105 generates and displays the content by traversing the data flow graph and executing drawing instructions as they are encountered in the graph.

Portions of the data flow graph are compiled (converted to machine code) by a compiler within player client 105, at run-time. Code is thus compiled on an as-needed basis. This compilation is redone in response to changing circumstances, such as when an optimization by recompilation would result in improved performance, so that only those portions of the code that are affected by the change are re-compiled. This improves efficiency of the compilation and content generation operations. For example, in one embodiment the system of the present invention performs such optimization by generating Span Engines in an efficient manner, as described in more detail below.

By storing a compiled representation of portions of the data flow graph, player client 105 in one embodiment is able to re-use previously compiled code without re-compiling.

Authentication

The present invention also provides a system for verifying the integrity of served content. In one embodiment, player client 105 performs an authentication check on each item of content before playing it. If the content fails the check, it is not played, thus preventing illicit content modifications (as well as protecting against transmission errors).

One technique for authenticating content is to derive a number from the content received from content server 102. If the number does not match a predetermined parameter or value, the authentication check has failed and the content is not displayed. Information describing the failed authentication check may be sent to data collection server 103.

An additional level of authentication may be provided for Authoring. While a piece of content is being Authored, it may be open to modification by a particular user (Author) or list of users (Authors). "Player locks" are provided in the form of authentication checks or a list of authorized player clients 105 to determine whether a particular player client 105 is authorized to modify a particular content item.

Only an authorized user having the authorable content file and having the appropriate Author rights can revise the content, and only an authorized user having Publish rights can republish it (though the Author and Publisher may be the same individual in two different roles).

One embodiment of the present invention provides a scheme for preventing unauthorized use of <embed> tags by unknown domains. When content is Published, a list of authorized domains is included in the .ofb file. When player client 105 obtains content from content server 102, it obtains the list of authorized domains and determines whether the domain of the web page containing the <embed> tag is among the listed authorized domains. The served content is displayed only if the domain is authorized; otherwise player client 105 does not play the content. In one embodiment, a warning message can be displayed if an unauthorized domain name is detected; in another embodiment, player client 105 can shut down in such a situation.

Personas

In one embodiment, the system of the present invention is able to identify each of a number of users of a particular client machine. Any number of "personas" can be identified and associated with a client machine, so that the content server can provide content appropriate to the currently active persona. This increases the likelihood that the end user will be interested in the content, and thereby improves advertising targeting.

In one embodiment, the system of the present invention is able to identify individual personas for different times of day, so that a user logging on in the morning, for example, might be assigned a different persona from a user logging on using the same machine in the evening. Player client 105 may observe that morning users tend to access certain types of sites which are business-related, while evening users tend to access entertainment-related sites. This may be an indication that at least two different users should be identified and given distinct personas.

Other methods of identifying personas may also be used. For example, the currently active profile may identify the particular user of the computer, or more sophisticated pattern recognition may be employed.

For each identified persona, a cluster relating to demographics, psychographics, or any other subset or categorization, is identified based on the associated user's behavior and interests. Player client 105 determines this information by observing the types of sites visited by the user.

Cluster information can then be provided by player client 105 when it makes a request for content from content server 102. Content server 102 selects appropriate content based on the specified cluster. In this manner, improved advertising targeting is accomplished, since each user is more likely to see ads relating to his or her interests.

Additional Personalization and Customization

In one embodiment, the present invention employs an additional designator that specifies the equipment type of the client machine. This designator identifies, for example, whether the client machine is a computer, cell phone, wireless PDA, or other type of device. Content server 102 is thus able to provide content that is appro-

priate for a particular type of machine, given its inherent characteristics and limitations. For example, content that is displayed on devices such as cell phones or wireless PDA's could be presented in a format that is specifically tailored to the small screen sizes and limited resolution of such devices.

In alternative embodiments, knowledge base 306 and/or data collection server 103 obtains and collects data describing the particular software and hardware characteristics of browser 104 and player client 105, so that appropriate content for the user's computing environment can be selected and served to player client 105.

Identifiers

In order to provide personalized content, and in order to track users and transactions, the system of the present invention employs a number of different identifiers, or "IDs". The following is a description of a set of identifiers and their relationships as implemented in one embodiment of the present invention. One skilled in the art will recognize that many other types and formats of identifiers are possible; therefore, the following description should not be considered as limiting the scope of the invention as claimed herein.

User ID (UID) – A unique 12-digit decimal number used to identify each user of the system.

Originator ID (OID) – A UID used to identify the Originator of a Project. Used in combination with a Project Number (PRN) to form a unique Project identifier (PRJ) when a new Project is opened.

Manager ID (MID) – A UID used to identify the Manager of a Project.

Author ID (AID) – A UID used to identify each Author of a Project.

Project Number (PRN) – A unique (per OID) 8-digit sequential decimal number used to identify projects for a specific originator (OID). Used in combination with an Originator ID (OID) to form a unique Project identifier (PRJ) when a new Project is opened.

Project ID (PRJ) – A unique 20-digit decimal number used to identify a Project. The Project (PRJ) is created by combining the Originator ID (OID) with a Project Number (PRN) when a new Project is first opened.

Publisher ID (PID) – A UID used to identify the publisher(s) of a Project. The PID is used in combination with a Publication Number (PBN) to uniquely identify a Publication (PUB).

Publication Number (PBN) - A unique (per PID) 8-digit sequential decimal number used to differentiate publications for a specific PID. The PBN is used in combination with a PID to provide a Publication ID (PUB) uniquely identifying a Publication.

Publication ID (PUB) – A unique 20-digit sequential decimal number used to identify a specific publication "space" that may include any number of Articles. The PUB is used in combination with the PRJ to uniquely identify an Article (ART).

Article ID (ART) – A unique 40-digit decimal number used to identify a specific published instance of a Project. The ART is created by combining the PRJ with the PUB when a Project is published.

Demographic ID (DID) – A unique 6-digit decimal number used to identify a specific group based on demographics, psychographics, or other factors or characterizations. The first two digits are reserved; the third and fourth digits identify the group (or cluster); and the fifth and sixth digits identify the subgroup (or specific group). The DID is used to request and deliver targeted publications.

Player ID or Serial Number (SN) – A unique 20-digit decimal number used to identify a player client. The SN is used to track end-user interaction within the system of the invention.

Right (RIT) - A unique three-digit binary number used to identify a permission that a UID has on a Project. The RITs include 000-Reviewer, 001-Manager, 010-Author, and 100-Publisher.

Role Number (RN) – A 1-digit decimal number used to identify the set of permissions that a UID has on a Project. The RN is calculated for a UID adding all the RITs for the UID. For example, a UID having the initial roles of Manager and Author would have an RN of 000+001+010=3 (decimal). For a UID that is also a

Publisher, the RN would be 000+001+010+100=7 (decimal). A UID with RIT=0 would be a Reviewer and could only view the project.

Role ID (RID)– A unique 13-digit decimal number used to define a UID's permissions. The RID is a combination of UID and the UID's RITs.

For example, a Project might have a Manager, two Authors, a Publisher, and an Author who is also a publisher. In addition, a role of Reader would apply to everyone else. The UIDs might be assigned sequentially. The RIT for each UID would be appended to the UID to form the RID, as follows:

	UID	RIT	RID
Manager	0000000002	001 = 1 (decimal)	00000000021
Author	0000000003	010 = 2 (decimal)	00000000032
Author	0000000004	010 = 2 (decimal)	00000000042
Publisher	0000000005	100 = 4 (decimal)	00000000054
Author & Publisher	0000000006	110 = 6 (decimal)	00000000056
Reader (everyone else)	0000000000	000 = 0 (decimal)	00000000000

Server ID (SID) – A unique 8-digit hexadecimal number used to identify a content server. In one embodiment, this ID is the same as the public IPv4 address of the content server.

Transaction Number (TN) – A unique number generated at content serve time.

Transaction ID (TID) – A unique Instance identifier that combines the SID and TN to uniquely identify a specific serve of an Article.

File Names and Path Names

In one embodiment, Projects are stored as files having a Project Filename of PRJ.ofb, where PRJ is the Project ID (20-digit decimal). When published, the Project becomes an Article. The filename is then prepended with the Publication ID (PUB)

and the PUB is inserted into the binary content file (extension .ofb), for data tracking purposes. Thus, Article Filenames take the form "ART.ofb" where ART is the Article ID = Publication ID (PUB) & Project ID (PRJ) (40-digit decimal).

A Publication can contain several different Articles. The PUB part of the ART would be identical for each Publication, but the PRJ would be different.

The publisher's URI uses the PUB alone. In this way, the player can request personalized content by obtaining a different Article for the same URI, depending on the supplied demographic ID. Each combination of a DID and PUB yields a path that is hard-linked to a particular file containing an Article. This is useful for multi-targeted campaigns in which advertising messages may be tailored to specific groups. For example, if there are two Articles in the same publication, each having an associated file, such as ART1 and ART2, DID2/PUB1 and DID6/PUB1 could both link to ART1, while DID0/PUB1, DID1/PUB1, DID3/PUB1, DID4/PUB1, and DID5/PUB1 could all link to ART2. In this way, a request for PUB1 returns different Articles, depending on the DID associated with the user. A similar technique is employed in one embodiment, for providing content tailored to specific client machine equipment types. An additional identifier is provided for specifying an equipment type, and the appropriate hard-links are established so as to serve content that is appropriate to the specified equipment type. For example, DID1/EQP1/PUB1 might link to ART2, which DID1/EQP2/PUB1 might link to ART3, where EQPx specifies the equipment type of the client machine. Thus, a client machine having relatively limited output and/or bandwidth capability (such as a cell phone) would be designated EQP2 and would receive a more limited version of the content (ART3), than would a client machine such as a desktop computer. In one embodiment, the specific limitations of each potential client machine are determined in advance so that appropriate content can be generated and available for each such machine.

In one embodiment, the appropriate DID and equipment identifier are stored by player client 105, and supplied as part of the request to content server 102. PUB1 is the URI the publisher utilizes to identify a particular publication. Thus, player client 105 controls which demographic (or psychographic, or other subset) link is requested without having to change the URI, and without requiring any conditional overhead on the part of content server 102 to process the request. This technique, in

effect, removes the burden of demographic decision-making from content server 102 and distributes it over all the player clients 105 and servers.

For a non-targeted campaign, all DIDx/EQPx/PUB1 paths would link to a single article, for example ART1, where ART1 typically corresponds to the only Article in this publication. All links point to the same Article. Thus, no matter which demographic or equipment type player client 105 specifies, the same Article is returned.

The above-described technique allows player client 105 to always request content the same way, regardless of whether the content is targeted or non-targeted. Differentiation takes place by controlling the links on content server 102. This reduces the decision-making overhead on player clients 105.

In one embodiment, player clients 105 are hard coded to request Demographic 0 and Equipment Type 0 by default. If no other Demographic and Equipment Type are specified, player client 105 will request the content for Demographic 0 and Equipment Type 0.

In the above description, targeting is described in terms of Demographics and Equipment Types. One skilled in the art will recognize that other subsets or clusters may be employed without departing from the spirit or essential characteristics of the present invention, such as for example psychographics or other subsets.

System Resources

Referring now to Fig. 3, there is shown a block diagram of a set of resources for practicing the present invention. In one embodiment, system resources are divided into three tiers: central resources 301, hosts and ISP resources 302, and end user resources 303. Central resources 301 are generally within the control of the entity providing the services of the present invention, or agents of the entity. Central resources 301 include, for example: object store 304 for storing and providing objects for authoring; author server 101 for servicing authors of content and effecting publication of content to content server 102; content server 102 for accepting and responding to requests for content; plug-in server 305 for providing files necessary for installation of player client 105; data collection server 103 for collecting transaction and user data for tracking purposes; and knowledge base 306 for storing collected tracking data.

Hosts and ISP resources 302 are generally outside the direct control of the entity providing the services of the present invention. Hosts and ISP resources 302 includes, for example: web server 107 for serving web pages and content in a conventional manner, and stub server 307 for serving the stub of player client 105.

End user resources 303 include resources that are under the control of the end user, such as browser 104 which runs on the user's workstation. Player client 105 is included as a plug-in for browser 104.

Referring now to Fig. 4, there is shown an example of an authoring operation in the context of the architecture of Fig. 3. In one embodiment, the authoring environment for creating content in connection with the present invention is accessed through player client 105, as described in more detail below. Player client 105 interacts directly with author server 101 and object store 304 to compose content to be played on player clients 105. Content is maintained in object store 304, but may be replicated to the end user's workstation for faster access, such as for developers with large environments or constrained bandwidth.

Referring now to Fig. 5, there is shown an example of a publishing operation in the context of the architecture of Fig. 3. In one embodiment, publishers interact with the system through web browser 104. A publisher creates a publication, fills it with articles, and sets targeting options on the article. When published, object store 304 "loads" (moves) the articles to content server 102, setting up the appropriate environment based on the details of the publication. The articles are verified, and a URI "embed" tag is delivered to the publisher for inclusion in hosted web pages (i.e. those supplied by web server 107).

Referring now to Fig. 6, there is shown an example of a content playing operation in the context of the architecture of Fig. 3. The block diagram of Fig. 6 shows a scheme for implementing the flow chart described above in connection with Fig. 2. If, when browser 104 detects 204 <embed> tag, it determines that player client 105 has not been installed, browser 104 contacts stub server 307 which provides the end user with a choice to download player client 105. Once the stub of player client 105 is delivered to browser 104, it contacts plug-in server 305 to serialize player client 105 with a unique Player Serial Number. Plug-in server 305 then delivers the rest of player client 105 and it is installed in browser 104.

Once a player client 105 is available, content server 102 provides content to be output by player client 105. If appropriate, the domain from which the content request was generated is authenticated against a stored list of domains. As described above, each content "serve" is serialized by a unique TID for tracking purposes. Player client 105 renders the content and displays output, and also begins to track user behavior, such as click count and visibility time. At some point, such as when the user closes the browser, player client 105 saves the tracking data locally (for fault tolerance) and sends a packet containing transaction and client data to data server 103. If the packet is not successfully transmitted, player client 105 attempts retransmission at some later point, such as when player client 105 is reactivated. In one embodiment, Port 80 and HTTP are used for data communication, to allow the transfer of data from any system with web access to the Internet.

Referring now to Fig. 7, there is shown an example of a data transfer operation in the context of the architecture of Fig. 3. Knowledge base 306 runs a data loader program (not shown) that retrieves server and player data from the appropriate server, either content server 102 or data server 103. The loader processes the data, checking it for referential integrity before moving the data to warehouse 306. In one embodiment, data is stored in three tables: a transaction table (for storing completed transactions suitable for billing to an advertiser), a server incomplete transaction table (for storing transactions that were incomplete at the server end), and a player incomplete transaction table (for storing transactions that were incomplete at the player client end).

The Player Client

The present invention facilitates the display of sophisticated content without requiring large amounts of data to be downloaded to the client browser. This is accomplished by performing much of the content generation processing at the client end, rather at the server end. A player client 105 is provided to perform this functionality.

Player client 105 is a web-based application that in one embodiment runs within a browser application 104. Player client 105 can be provided as an add-in or plug-in module for browser 104, or may be included as a component of browser 104

or even as a stand-alone software application. Player client 105 may operate in any of a number of hardware environments, including a conventional computer connected to the Internet, a cell phone, a PDA, and the like. As a browser plug-in, player client 105 is activated when an appropriate mime-type is encountered by browser 104, as is known in the art. A particular file extension, e.g. ".ofb", can also be used to trigger browser 104 to activate player client 105. When such content is encountered and player client 105 activated, browser 104 passes the .ofb file to player client 105 which is responsible for playing the content.

In one embodiment, player client 105 is freely downloadable from a web site. In another embodiment, a "stub" is distributed, either separately or as part of browser application 104. The stub is a small application which is activated when browser 104 is run. The stub, when activated, makes contact with stub server 307, or some other server, which provides the latest version of player client 105 in response to a request from the stub.

In one embodiment, a serial number is assigned to each instance of player client 105 when it is downloaded to a client machine and installed in browser 104. This unique serial number, also known as a Player ID, identifies the particular player client 105, and is used to track end-user interaction within the system of the present invention. In one embodiment, the serial number is a 20-digit decimal number.

In one embodiment, the stub identifies one of several mirror sites (based on geographical factors, for example) from which to download player client 105. The downloaded player client 105, once installed, is able to keep track of which mirror site was its source. This selection of a mirror site can then be used to help select a source for content when requested.

Player client 105 is capable of delivering high-performance graphics, sound, and interactivity that go beyond the capabilities of conventional web browsers. Player client 105 runs scripts received from content server 102 to generate content.

Data Flow Architecture

In order to provide functionality for generating premium content and interactivity without requiring large downloads or excessive bandwidth, player client 105 in one embodiment is implemented using an object-oriented data flow architecture that ensures maximum flexibility and scalability of function, including

ensures maximum flexibility and scalability of function, including both authoring capabilities and content display capabilities. The implementation of data flow architecture of the present invention facilitates improved modularity in the construction and operation of the software of the present invention.

In a data flow architecture, data (including images, sounds, animations, and the like) are transformed, processed, and displayed by passing them through a particular machine that is designed to produce the desired result. The machine is implemented as software that is constructed from components called nodes. The nodes are connected with one another in a particular topology to implement the desired functionality. This interconnectedness and modularity allows player client 105 and associated software systems of the present invention to be implemented with a great deal of flexibility, so that a machine for generating any desired result can be easily constructed.

A particular machine's configuration, including the nodes, their connections, and the overall topology of the machine, can be represented in a data flow graph, as is known in the art (also referred to as a directed graph).

In one embodiment, data flow graphs of the present invention are acyclic in nature, so that traversal of the connections within a graph will always eventually end in a node have no outgoing connections (i.e., a leaf node). Ensuring that data flow graphs are acyclic avoids the possibility of infinite loops during execution.

Referring now to Fig. 18, there is shown an example of a simple content element implemented using a data flow graph. Data flow graph 1800 includes four nodes which combine to display a bouncing ball. Bounce node 1801 specifies the movement associated with a bounce. Ball 1802 specifies the appearance of the ball, including for example color, size, texture, and the like. Number 1803 contains a value that is used as a parameter of the bounce effect, such as for example the speed of the bounce. Frame buffer 1804 is the output channel for displaying the content.

When authoring, number 1803 can be replaced by a slider control (not shown) to adjust the value of the parameter supplied to bounce 1801. The user can specify that this slider control be removed and replaced by a fixed value when the content is published. Alternatively, the slider control can remain in the content, if desired, so that the end user can control the speed of the bouncing ball (or any other parameter).

The Author specifies which nodes (such as the slider) are to be replaced by fixed values or stripped out when the content is published.

If the Author is using an authoring environment, as described below, certain controls (such as authoring sliders) are by default removed upon publication. The Author can change this if desired.

Execution of a data flow graph is implemented as follows. Execution begins at a display or output node (such as frame buffer 1804) and proceeds recursively through the graph following each connection in turn. During this traversal through the graph, information is collected from each visited node into a data structure denoted "CurrentState". Such information includes, for example, position, orientation, compositing mode, contrast volume, and the like. When a leaf node (such as ball 1802 or number 1803) is finally reached, CurrentState is finalized.

As the graph is traversed, the system keeps track of a node reference called the "context node". Initially, the context node is the display node, but this may change in the course of the graph traversal. Thus, the context node is a temporary backward reference that is developed at run-time and that is not explicitly set forth in the graph topology. The leaf node sends a message to the context node to have its image or sound output into the context node.

The context node uses all the information collected in CurrentState to perform the drawing operation to output the content.

Homogeneous Objects and Polymorphism

In one embodiment, the data flow architecture of the present invention is implemented using an object-oriented approach. Nodes correspond with objects, and the connections among nodes are represented by object references held by nodes. Data flows from each parent node through the connection to a child node, by sending a message to the child object, as is known in the art of object-oriented implementation.

In the present invention, flexibility is maximized by allowing a node to connect to any other node. Thus, every node is able to receive any message sent by any possible parent. Enforcing such polymorphism among nodes, and assuring that the system as a whole consists of nodes belonging to a single class, ensures that all nodes

speak the same language and can be implemented using a single Application Programming Interface (API) supported by all nodes, and further ensures that any node can be connected to any other node without causing errors or crashes. This scheme is referred to as a homogeneous object system.

Nodes can be connected with one another as many times as desired, with no theoretical limits to the complexity of the data flow graph. Subsections of the data flow graph can be thought of as "super-nodes", or "modules", which act like normal nodes but are implemented as sub-graphs of nodes. A super-node can be employed anywhere in a data flow graph in place of an ordinary node.

Furthermore, a homogeneous object system as employed in the present invention permits inclusion and removal of user interface elements as needed. Thus, such elements can be included for authoring purposes, and replaced by fixed values (or removed altogether) when content is published. Homogeneity insures that such automated transformations to the graph topology will work properly.

In addition, the data flow graph architecture of the present invention allows authoring at any level of abstraction. An author can create a piece of content that is itself an authoring tool. This can continue to any desired degree or level. Thus, authors are empowered to create sophisticated content in a streamlined and efficient manner.

The above-described architecture, including the homogeneity of the data flow graph, thus facilitates a system in which authoring interactive content and playing the published content are essentially the same operation, with selected nodes being replaced and substituted as desired.

In one embodiment, the data flow model of the present invention allows information to flow in both directions for every connection in the data flow graph.

Finally, the above-described data flow model facilitates construction of a component-based authoring system wherein a large number of content elements are made available for inclusion in a particular piece of content. These content elements can be provided on a centralized object store 304 so that they are available to Authors when needed, but can be updated and enhanced, and new elements added, as desired.

Each content element, or component, contains its own user interface if applicable, which can be stripped out when the content is published. This modular approach, and the stripping out of UI elements, is facilitated by the homogeneity of the data flow graph.

The data flow graph of the present invention, implemented in a homogeneous approach, thus provides a fundamentally novel mechanism and architecture for the creation and display of interactive content. One skilled in the art will recognize that such a technique may be applied in other embodiments and implementations of software design and construction tools.

Types of Nodes and Attributes

A Null node is provided that can receive all messages, but simply does nothing. In accordance with the polymorphism of the system, the Null node can be placed anywhere in the data flow graph without crashing the system or causing an error. This improves upon conventional schemes, in which a null object reference consists of a pointer set to zero which, if it used, will generally crash the program. In one embodiment of the present invention, Null nodes do not crash the program, but merely do nothing when encountered.

Other types of nodes may be made available according to the particular needs of the application. Examples of nodes available in one embodiment include:

- **Control Nodes**: Control collections of nodes. Examples include: Document, Layer Manager (for compositing layers), and ParticleSystem (for generating and controlling particles).
- **Image Nodes**: Manage image memory. Examples include: RGBACache (for holding color pixels), MaskCache (for holding single 8-bit number pixels), and CoordCache (for holding pixels that are two-dimensional vectors).
- **Effect Nodes**: Alter the way images are displayed or sounds are played. Examples include: Contrast (for changing color and brightness of drawn pixels), Volume (for changing sound volume), Warper (for performing warping operations), Masker (for performing masking operations), and Blur (for causing underlying images to be blurred).

- **Math Nodes**: Examples include: Scalar, Vector, and Color; add, subtract, multiply, and divide.
- **Procedural Image Creator Nodes**: Generate image elements. Examples include: GradientImage (for filling a rectangular array of pixels with an animated color), RadialBlend (for creating rings from a single row of pixels), SweepBlend (for creating rays from a single row of pixels), Path (for rendering a PostScript path with anti-aliasing), and Text (for allowing font outlines to be rendered as paths with text).
- **Animation Nodes**: Generate animation elements. Examples include: Interval (for creating a linear path from two points), Bezier (for creating a spline path from four points), Sequence (for allowing multiple animations to be sequenced in time and played), Rotation (for appending a full three-dimensional rotation to the current transformation matrix), AxisScale (for appending a three-dimensional scale to the current transformation matrix), Affine (for rendering an object with the transformation matrix), and PerspectiveScale (for setting a uniform perspective-based scale factor).
- **Interactive Nodes**: UI elements or controls in interactive content. Examples include: Button, Drag, Slider, and Dial.
- **Sound Nodes**: Implement sound functionality. Examples include: SoundCache, and SoundControl procedural sounds.

One skilled in the art will recognize that many other nodes and node categories are possible.

The present invention also provides attributes, in the form of named connections among nodes. Thus, a particular node views its set of connections as attributes that are used by the node to perform its intended functionality. For example, the Masker node employs a mask and an image that are provided via node connections and thus act as attributes for the Masker node. In one embodiment, each node has a set of attributes it supports. Attempts to connect a node via a non-supported attribute results in no connection being made.

Examples of attributes include:

- Raw pixel data;
- Sources of pixel data (other nodes);

- Color components;
- Animated values;
- Animation controls;
- Time controls;
- Selectors; and
- Path descriptors.

One skilled in the art will recognize that many other attributes are possible.

Scripting Language

In one embodiment, the data flow graph is specified using a scripting language. In another embodiment, an authoring environment is used for the creation of content, which is then represented as a series of interconnected nodes. In yet another embodiment, either the scripting language or the authoring environment may be used, depending on the complexity of the content to be created, the level of expertise of the Author, and other factors.

In general, in the present invention, data flow graphs are alternative representations of scripts, and the two forms of representation have a one-to-one correspondence. Thus, a script can be converted to a data flow graph, and vice versa, with no loss of functionality or information.

For example, if a scripting language is used, the following syntax may be used for specifying nodes and attributes:

```
<NodeName
     <! Attributes go here
/NodeName>
```

< NodeName begins the section describing the attributes of a new node 'NodeName'. The closing construct **/Name>** indicates the end of the section. As in XML, nodes may be defined within other nodes; everything within the node definition for NodeName will be affected by any surrounding nodes.

The node definitions created using the scripting language define the data flow graph. Each defined node corresponds to a graph node, and the attributes of nodes

define the connections in the data flow graph. Generally, when a node is created (i.e. defined), all its connections or attributes are specified, using the syntax shown above.

One skilled in the art will recognize that the hierarchical syntax of the above-described scripting language can only create tree-like structures, not generalized acyclic graphs. The present invention thus provides capability for nameable objects, in order to extend the graph topology to allow any acyclic configuration. While tree-like structures can, in general, branch in an outward direction only, an acyclic graph can also re-join itself (without circling back on itself). The ability to name objects allows the system of the present invention to provide multiple references for a named object in any number of places in the graph, so that the tree can join on multiple references to any particular node, and therefore become an arbitrary acyclic graph.

An example of a node with attributes in the context of a scripting language is as follows:

```
<NewNode
     NumberAttribute = 1.2
     VectorAttribute = (1,2,3)
     ColorAttribute = [.3,.4,.5,1]
     SelectorAttribute = SelectThis
     ChildAttribute = <AnotherNode
          ChildsAttribute = Whatever
     /AnotherNode>
/NewNode>
```

In the syntax of the above example, attributes are assigned to a node using an equal sign. In one embodiment, node names are distinct from attribute names, so that nodes and attributes can be differentiated from one another both by syntax and by the name itself.

For clarity, the structure of the script follows the topology of the corresponding data flow graph. Nodes can be nested as many levels deep as desired.

Names can be assigned to elements of the data flow graph as follows:

```
<OurNode
     ID = fred
```

```
/OurNode>
<TheirNode
     Attribute1 = fred
     Attribute2 = fred
/TheirNode>
```

The ID Attribute causes the User Name `fred` to be associated with that specific instance of OurNode. TheirNode uses `fred` for two attributes. When `fred` is used, the entire sub-tree is referenced inside OurNode. By using referencing instead of making a separate copy, memory space is conserved.

Scripts are converted from ASCII to a binary file having a distinct extension such as ".ofb". Player client 105 reads the .ofb file and constructs a data flow graph based on the script. The data flow graph operates as a virtual machine to produce content, including visual elements, sounds, and interactivity.

Span Engines

To improve the efficiency and performance for operations involving the generation and display of graphics, the present invention employs a particular type of subroutine, called a Span Engine, for performing graphics operations and sound operations. Each Span Engine applies a specific operation to a contiguous row of pixels or sound samples. By implementing Span Engines in generated machine code, the present invention greatly improves the performance of player client 105 in rendering graphics. In addition, all generated machine code can be confined to the Span Engines, thus avoiding the maintenance and compatibility problems associated with widespread use of pre-written assembly language.

For improved efficiency, specialized Span Engines are provided for different situations. For example, one Span Engine may be provided for moving a sprite, while another Span Engine may be provided for a combination of moving and dissolving a sprite. Though such specialization may result in a large number of Span Engines, the individual Span Engines are capable of extremely fast performance.

In one embodiment, in order to reduce the size of player client 105, the Span Engines are built as needed, rather than being provided in precompiled form within player client 105. Player client 105 contains a compiler which detects when a new

Span Engine is needed and builds (compiles) the Span Engine for the situation at hand. Span Engines can be compiled extremely quickly, due to their small size, so that player client 105 can perform this operation during an animation, if needed. The present invention thus provides a technique for call time, on-demand, situation-optimized compiling to native machine code.

In general, only a small number of Span Engines are needed at any given time, so that player client 105 is able to render content without growing to an unduly large size.

After traversing part of the data flow graph and collecting the information in the CurrentState structure, player client 105 examines the elements of the structure and determines which span engine to invoke based on the values in the structure. In one embodiment, ten variables are used as indices to a ten-dimensional array of Span Engine pointers. If a value is non-zero, it refers to a valid Span Engine; otherwise a new Span Engine is created and inserted in the array. In another embodiment, a hash table is created based on the values of ten variables, and a lookup is performed into a smaller table based on the hash results.

In general, most scripts can be handled by a relatively small quantity of Span Engines. The array implementation can be used when a small quantity is required, but if the player client 105 would cause the user's computer to run out of memory, the hash table implementation may be more feasible.

In one embodiment, all Span Engines are of the same form. In particular, they may be embodied as a single loop that operates on a contiguous row of pixels, regardless of the particular type of information in each pixel. They can operate on one or two sources to generate output for one destination. One example of pseudo-code for a Span Engine is as follows:

1. Compute address of first pixel in destination
2. Compute address of first pixel in source 1
3. Compute address of first pixel in source 2 (if there are two sources)
4. Compute number of counts N in the loop; assign counter to value N
5. For each pixel:
   - Load source value (using address)
   - Perform an operation on the values

- Store the values in the destination
- Move the destination address to the next pixel in each case
- Decrement counter; repeat step 5 if counter is greater than zero

The compiler operates as follows. When a Span Engine is needed, the compiler is invoked with the desired variable state. The compiler operates by concatenating a series of small code-fragments to generate a new Span Engine. The fragments themselves are selected according to the variable state. By providing these individual fragments within player client 105, and allowing player client 105 to compile by joining fragments into Span Engines as needed, the present invention provides an effective mechanism for implementing a large number of Span Engines in a small amount of memory.

In addition, such a scheme allows for expandability as well. Additional fragments can be provided and integrated more easily, and improved flexibility can thus be provided.

Machine Code Generation

To accomplish a high level of performance, in one embodiment player client 105 generates machine code "on the fly" for implementing performance-intensive sections of the code. The use of generated machine code also permits code in player client 105 to take advantage of high-performance processor instructions such as MMX and Alti-Vec, which can further improve performance.

The compiler of the present invention compiles to pure machine code, in one embodiment. The compiled code is therefore readable by the microprocessor itself, with no additional interpretation being required for the code to run.

The compiler of the present invention occupies a relatively small amount of memory because it re-uses code fragments to create required Span Engines.

Animation

The present invention enables broadcast-quality content without requiring large amounts of data transfer. This is accomplished by using animation to simulate a video experience, while requiring a minimum of transactional load between the server and client. Furthermore, unlike broadcast media, the content presented by the present invention is fully interactive; by employing homogeneous objects as de-

scribed above, the present invention is able to substitute an interactive element for any animated component or control. Since player client 105 is capable of compiling and executing code on the fly, it can display animated content, simulating a broadcast video experience, in a highly efficient manner.

In one embodiment, player client 105 accomplishes many effects using the generalized concept of animation. Such effects include, for example, gradients, blurring kernels, and lookup tables. By implementing many related features with a single subsystem, player client 105 is kept highly efficient and small.

The present invention generalizes the concept of animation beyond its normal definition of changing an element or parameter over time. In the present invention, the element or parameter can change depending on an indication of spatial position, dimension, or other factor. Thus, in the following discussion, where the independent variable is referred to as "time" for illustrative purposes, other independent variables such as spatial position or dimension can be substituted.

Several nodes make use of the animation functionality and waveforms, such as for example:

- <Interval animate between two values
- <Bezier animate four values using Bezier spline interpolation.
- <Cardinal animate four values using Cardinal spline interpolation
- <FlipBook animate N frames
- <Sequence animate N sequential animations

In one embodiment, player client 105 (and the authoring environment) include built-in waveforms that can be used to author and present content. Waveforms may be used for interpolating between various control points or objects. In one embodiment, each waveform is controlled by up to four parameter attributes.

Referring now to Fig. 16, there is shown an example of a graph 1600 including nonperiodic waveform 1601 as may be used in a typical animation. A value increases from a starting point of 0.0 to an ending point of 1.0. This value can be used to control any function, such as an output characteristic of the content. Four parameters 1602-1605 are supplied to define the shape of waveform 1601. The interval of time between Begin 1602 and End 1605 defines where the waveform is active, for nonperiodic waveforms. In 1603 and Out 1604 further define the shape of waveform 1601 and

control the slope of various portions of the curve. From Begin 1602 to In 1603, the value increases at an accelerating rate. From In 1603 to Out 1604 the value increases at a constant rate. From Out 1604 to End 1605 the value increases at a decelerating rate.

Referring now to Fig. 17, there is shown an example of a graph 1700 including periodic waveform 1701 as may be used in a typical animation. By employing periodic waveforms such as that shown in Fig. 17, the system of the present invention is able to generate animations that repeat themselves. In one embodiment, a parameter called Period is specified; the part of waveform 1701 from zero to the end of Period 1706 is repeated. If Period is given a value of zero, the waveform is defined as nonperiodic. In the example of Fig. 17, additional parameters 1702-1705 are specified, defining the shape of waveform 1701.

Other parameters or attributes may also be provided. For example, a Phase parameter can be supplied, which is added to each of the time parameters in a waveform, shifting the waveform in time, whether it is periodic or nonperiodic.

In addition, any of the parameters can themselves be controlled by other waveforms, thus giving rise to additional flexibility.

Waveform type can be selected using an Animus attribute. Examples of values for the Animus attribute include:

- EaseInOut (transition)
- Spline (transition)
- Decay (transition)
- DecayOscillate (transition)
- Squishwave (transition)
- Pulse (periodic)
- Oscillate (periodic)
- Sawtooth (periodic)
- Round (periodic)
- Perlin. (irregular/random)
- Uniform (irregular/random)
- Poisson (irregular/random)

- Gaussian (irregular/random)

Authoring Environment

In one embodiment, the authoring environment for the present invention operates as a plug-in for web browser 104, or as a built-in feature of browser 104, in a manner similar to player client 105. In fact, the authoring environment may be implemented in the same software package as player client 105, but with additional features. Alternatively, player client 105 can be implemented as a subset of the functionality of the authoring environment. The authoring environment is a true What-You-See-Is-What-You-Get (WYSIWYG) system; the Author need not be aware of the underlying data flow graph that is created as he or she develops content.

The authoring environment generates editable content in the form of the data flow graph. The Author edits the resultant data flow graph by manipulating internal UI controls. This allows the Author to, for example:

- Add or delete nodes or super-nodes
- Adjust parameter values
- Change colors
- Select animations and areas of interactivity

The authoring environment writes a content file in the form of a description of the data flow graph, either by saving to disk or by sending the file to author server 101. When the file is later referenced, all settings and elements are retrieved and may be edited.

When the content is published, certain changes are made to the content file, as described below.

In the present invention, published content has the capability of being fully interactive. Thus, the authoring environment, including the tools associated with generating and modifying content, can be implemented as an additional level of interactivity that is simply removed, or hidden, when the content is viewed by an end user. The content can thus contain its own user interface for performing adjusting and modifying operations. This user interface is visible only when the content is viewed using an authoring environment, but is not visible when viewed using the player cli-

ent 105. As discussed above, this feature is enabled in one embodiment by the use of a data flow graph employing object homogeneity.

In another embodiment, the user interface is removed from the content when the content is Published. Other interactive components intended for the end user may remain available. Such components may provide functionality for physical interactivity as well as logical interactivity. Physical interactivity includes the capability to change the appearance of the content based on user actions, such as for example changing a color in response to a user manipulating a slider control. Logical interactivity includes the capability to change deeper levels of behavior of the content, such as for example the destinations of certain links, based on the user's actions. Thus, the present invention is capable of displaying various levels and varieties of content based on interactions, user profiles, and sophisticated decision rules.

Alternatively, where player client 105 and the authoring environment are implemented as equivalents to one another, the user interface can be included in the published content, but may not always be visible. The visibility or invisibility of the user interface can be determined by the permission rights of the user. In other words, when the content is viewed by an Author, the user interface becomes available, but when the content is viewed by anyone else, the user interface is hidden.

In essence, the present invention removes the distinction between user interface components and content elements, so as to improve flexibility and simplify operations.

In one embodiment, the authoring environment of the present invention makes use of "super nodes", as described above, to allow the system to be broken down into hundreds or thousands of separate components that authors can assemble to create content.

Referring now to Fig. 9, there is shown an example of a screen 900 containing an authoring environment provided by the present invention. A number of layers 901 are shown, representing overlapping authorable graphics that are overlaid to form the final image. The user can click on any of layers 901 to perform editing operations thereon, and can add more layers as needed. Authorable content 902 represents the content being authored, and in one embodiment plays in real-time during the authoring process, including any animated or interactive elements. Controls 903

provide a mechanism for manipulating various parameters of content 902, such as for example color values, speed of moving elements, and the like. The user can change values for these parameters by moving knobs along sliders in controls 903. Grid 904 provides solutions, internal UI controls, and other elements that the user may activate or include in content 902 as desired.

Referring now to Fig. 10, there is shown an example of content 1000 that can be customized by an Author and branded or sponsored before publication. Interactive elements 1005 are included in content 1000, so that in effect content 1000 forms a web-based application when published. Content 1000 may be served on a website, or distributed through HTML e-mail, or offered to other sites as desired. Authoring buttons 1001-1004 are provided, allowing an Author to perform customization operations on content 1000 prior to publication. In the example, an Author can select a color theme for content 1000, add a banner and some pictures, and send content 1000 to publishing, by activating various buttons 1001. When published, buttons 1001-1004 will not be included in content 1000.

Referring now to Fig. 11, there is shown an example of a customization operation for content 1000. Once the user activates button 1001, window 1101 is presented, which allows selection from a number of color themes 1102. The user can select the desired color theme 1102, and the selected theme 1102 is then applied to content 1000. One skilled in the art will recognize that many other types of customization can be provided within the context of the authoring environment of the present invention.

Referring now to Fig. 12, there is shown an example of a user's project screen 1200 where previously authored projects are stored. Here, the user has activated button 1202, which is one way to evoke the project list to select and add a banner to a project the user is working on. Using screen 1200, the user can select any of a number of previously-produced banners (or other content elements) to add to the current project.

The user can select any previously generated project or content element from the project list of Fig. 12, for further modification or review. Pre-publish button 1201 opens the selected project in a preview area (not shown) where the user can preview, publish, or edit the project. Selecting the "edit" option loads the project in a dynamic

authoring mode, complete with all of the associated user interface authoring tools used to produce the project.

Publish button 1202 has an active state that indicates that the project has been published. Pushing button 1202 accesses a report sheet displaying the status of the project as well as tracking information such as number of views, click-throughs, and average time viewed. In one embodiment, this report sheet is only available to a user having Manager rights. For each project in the project list, a title 1203, thumbnail view 1204, and comments 1205 are provided.

Referring now to Fig. 13, there is shown an example of a sheet 1300 containing elements for project creation, arranged in a series of grids according to one embodiment of the present invention. In one embodiment, each solution on a sheet is identified through a series of database fields. For example, a "Pyrotechnics" sheet might contain: five animated flame types; four fireworks examples; three animated glowing backgrounds; four types of animated sparkles; three burning type effects; four smoke effects; and three magma textures. These elements can be applied as desired to content being authored.

My Projects button 1301 access project screen 1200 as described above in connection with Fig. 12. Theme Sheets button 1302 activates a floating window containing icons representing various sheets that may be selected. A series of sheet icons 1303 is presented in rows; activation of a sheet icon 1303 displays the elements of the corresponding sheet in grid 1304. Search button 1305 folds down a list of field names in a pull-down menu so that the user may search for specific elements and sheets by field names.

In one embodiment, each sheet is represented by a series of identification fields, including for example a number, name, and category. Particular elements on sheets have identifier numbers that indicate their affiliation with and position on the corresponding sheet. Elements also have identification fields specifying, for example, type of effect or tool, color, download size, CPU usage, and the like. In search field 1309, the user may enter query parameters based on any of these identification fields to locate elements matching the user's requirements. Results of the search are viewed as elements on a custom sheet, with the positioning of the results being determined based on an identifier number or degree of matching with the user's specified pa-

rameters. If a large number of matches are found, a series of tabs may be presented to allow access to various subsets of elements.

The user presses button 1306 to create a custom sheet. The user is prompted for a sheet name, and may further select an icon to represent the new sheet. Search field window 1310 allows the user to select the icon and highlight identification fields for classifying the new sheet for purposes of future searches.

Elements can be added to the currently-selected or new custom sheet by dragging any icon representing an element onto button 1306, which serves as a drop target for the new custom sheet. Preview button 1308 toggles from the currently selected sheet to the new sheet and shows the new elements added by the user.

An element from a custom sheet can be deleted by dragging the associated icon to delete button 1307. An entire custom sheet, including its contents, can be deleted by dragging the custom sheet's icon to delete button 1307. In one embodiment, a confirmation dialog box is presented before deletion occurs.

Referring now to Fig. 14, there is shown a control palette 1402 for a content element 1400. Each content element 1400 in a sheet 1300 has a control palette appropriate for the type of content included in the content element. In the example shown, control palette 1402 contains four slider controls 1401 for adjusting animated elements within content element 1400.

Referring now to Fig. 15, there is shown an example of hint help 1501 for control palette 1402. In one embodiment, hint help 1501 is displayed whenever the user holds an on-screen cursor over the associated control for a predetermined length of time. In another embodiment, hint help 1501 is displayed when explicitly requested by the user. Hint help 1501 provides assistance in operating the associated control element, such as a slider 1401, for control palette 1402.

In one embodiment, internal UI controls such as control palettes, control elements, and hint help are encapsulated in the element definition for each element of each sheet 1300. These internal UI controls are provided by author server 101 and are loaded dynamically when needed.

Publishing

When the Author specifies that the content is to be published, the following changes are made to the content file:

- Elements that are not to be included in the published version (such as authoring UI elements) are stripped out
- Authoring locks (if present) are removed
- Project Filename and Publication ID are assigned and attached
- Targeting information is added
- Unused content is removed
- Content is optimized for improved performance, bandwidth, and memory usage.

In a "load" operation, the content is then uploaded to content server 102, appropriate file links are created, and the load is verified. An <embed> code (including the correct URI) for accessing the content is made available to the Client, for inclusion on a web page. The <embed> code, when retrieved by an end user, will cause player client 105 to contact content server 102 and download the content as appropriate. In addition, a list of authorized domains can be provided and included in the published content file, so that any request for content can be verified as originating from an authorized domain.

Targeting information for the published content may be provided by an Author or Manager at the time of publication. In one embodiment, the Author or Manager fills out an online form specifying the variations of content that should be made available to different demographic groups, as well as to different equipment types, if applicable. The information provided via the online form is stored in content server 102; in one embodiment, hard-links are established between file paths and content files, as described above.

Installation

If the player client 105 and authoring environment are provided as a plug-in, the software may be downloaded from a website and installed automatically. Referring now to Fig. 8, there is shown a block diagram illustrating a site structure for downloading player client 105, including an authoring environment, according to one

embodiment of the present invention. This site structure may be implemented, for example, as a series of interconnected web pages providing access to introductory information, downloadable software for installing player client 105, and reviewing and publishing functionality. One skilled in the art will recognize that many other site structures and systems could be implemented for offering, installing, and using player client 105 and the authoring environment.

Home page 801 provides an entry page to the website with, for example, introductory information and links to a player client 105 download area. If a user follows a link to a download area, Get Player page 802 provides download information and acts as a registration area for users interested in obtaining player client 105. In one embodiment, stub server 307 transmits a stub installation to the user's workstation. When installed, the stub communicates with data server 103 to register player client 105 and to obtain the rest of the software needed to run player client 105. When an authoring environment is installed, additional features are activated in player client 105 to enable authoring. In one embodiment, such additional features are not available to the user until author registration has taken place, as described below.

The user then accesses Entry page 803 and Overview page 804, which explain how the system works and describe additional incentives and advantages of the system. Gallery 805 provides examples of content created by the authoring environment, including for example advertising and entertainment offerings. Users are invited to sample authoring their own content.

Trial Author page 806 allows a user to manipulate a sample advertisement as an introduction to authoring functionality. Limited functionality may be offered here, so that users are encouraged to register to obtain complete functionality. Author Overview page 807 provides additional explanation of the authoring system. Author Registration page 808 provides forms for the user to supply identifying information to register as an Author, and provides a password and other account information.

Author Tool page 809 provides the complete authoring environment for use by the Author, including various tools for creating splash screens, ads of various sizes, web-based applications, and the like. In one embodiment, the authoring environment

is identical to the content-viewing environment provided by player client 105, but with additional functionality in the form of various controls that may be used to manipulate the content. In other words, the final, published content, is a subset of the content as seen by an Author.

Client Registration page 811 provides an area for advertisers and ad brokers to set up a payment account and receive registration numbers for authorizing content publication. This area is used by "Clients", who may be Authors or may be separate entities responsible for payment for advertising.

Review/Publish page 810 allows Authors and Clients to view content, without the Authoring tools, before it is Published. This provides an opportunity for previewing the content as it will be seen by end users. If desired, the Author can allow content to be reviewed by others, such as co-Authors, prior to publication.

After publication, a complete version of the content, including the authoring user interface, is available for use by the Author in case he or she wishes to revise and republish the content.

Tracking page 812 provides tracking information for published content, such as for example the number of times each Article has been served, the click-through rate, the amount of time the Article was visible on a user's screen, and the like. In one embodiment, Tracking page 812 is only available to Clients and is password-protected to prevent unauthorized access. Information on Tracking page 812 may be obtained from knowledge base 306.

Billing page 813 provides information regarding billing for advertising and other content. This page may include, for example, billable rates for each published Article (based on tracking information), and the total amount charged to the Client. Functionality for paying the amount, such as via a credit card, may be provided. In one embodiment, Billing page 813 is only available to Clients and is password-protected to prevent unauthorized access.

Conclusion

The above-described invention thus provides a method, system, and computer program product for providing an authoring, generation, delivery, authentication, and tracking for online content such as advertising. The above description provides

merely exemplary embodiments for practicing the present invention. Those skilled in the art will recognize that other embodiments are possible without departing from
1490 the spirit or essential characteristics of the invention claimed herein.

Claims

What is claimed is:

1. A system for delivering and tracking content over a network, comprising:

a network;

a player client, coupled to the network, for requesting content over the network and for displaying received content;

a content server, coupled to the network, for transmitting content to a player client in response to a content request, and for transmitting tracking information describing content display and user interaction; and

a data collection server, coupled to the network, for receiving and storing tracking information.

2. The system of claim 1, wherein the player client transmits demographic information to the content server, and wherein the content server selects content responsive to the demographic information and transmits selected content to the player client.

3. The system of claim 1, wherein the player client transmits client information to the content server, and wherein the content server selects content responsive to the client information and transmits selected content to the player client.

4. The system of claim 1, wherein the player client transmits identifying information to the content server, and wherein the content server selects content responsive to the identifying information and transmits selected content to the player client.

5. The system of claim 1, wherein the player client is a plug-in in a browser application.

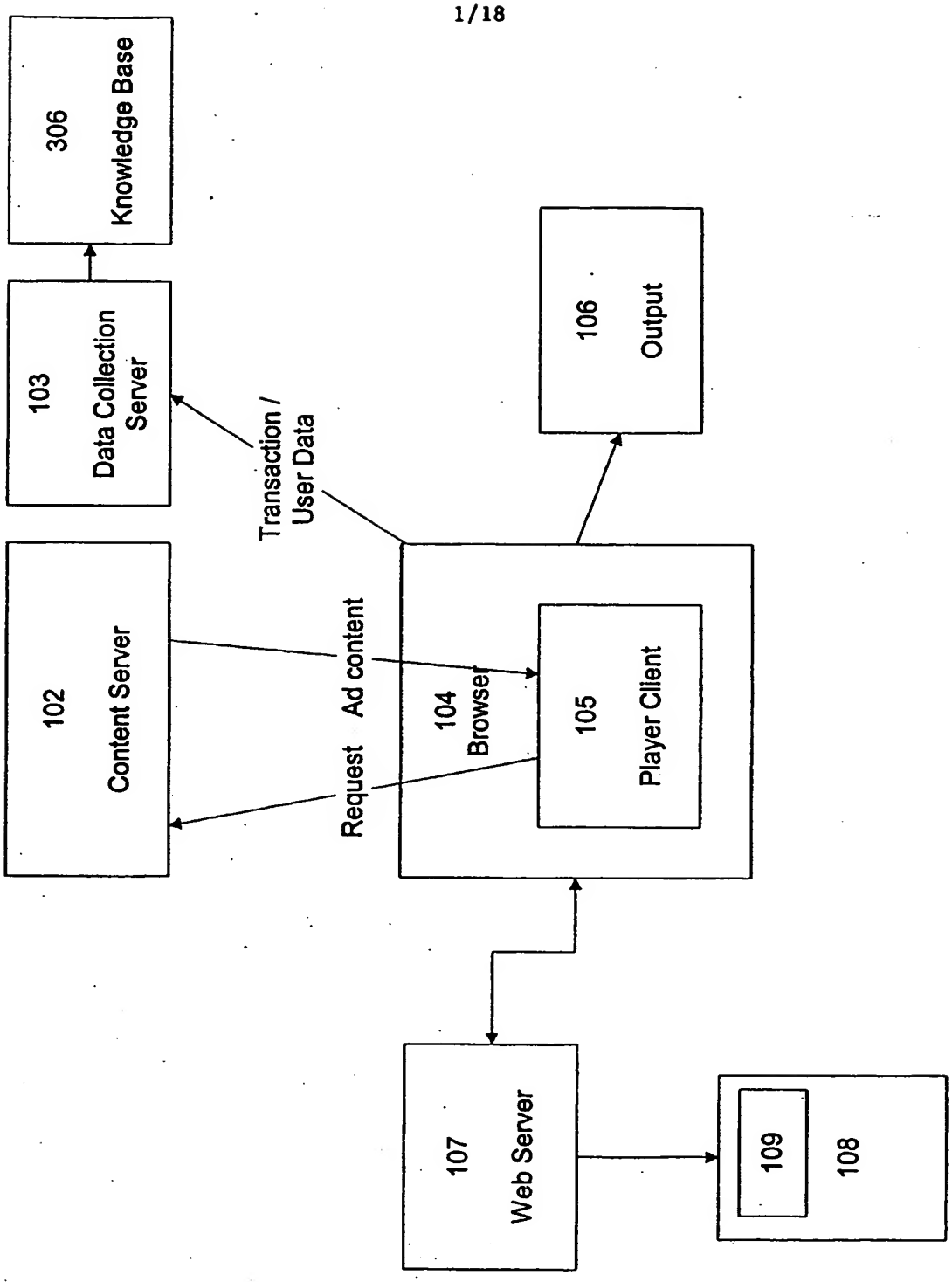

306
Knowledge Base
103
Data Collection Server
Transaction / User Data
106
Output
102
Content Server
Ad content
Request
104
Browser
105
Player Client
107
Web Server
109
108

FIG.1

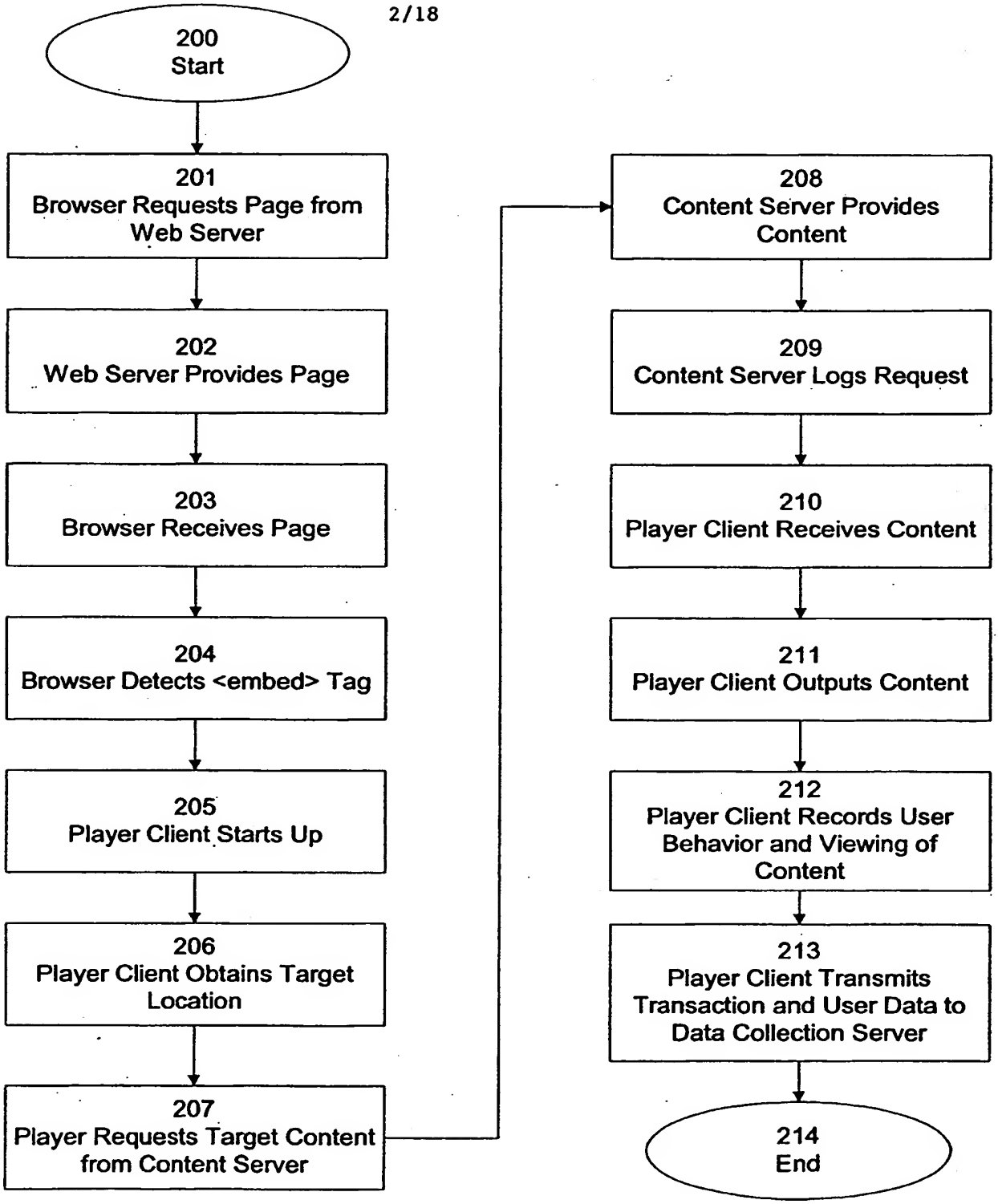

200
Start
201
Browser Requests Page from Web Server
202
Web Server Provides Page
203
Browser Receives Page
204
Browser Detects <embed> Tag
205
Player Client Starts Up
206
Player Client Obtains Target Location
207
Player Requests Target Content from Content Server
208
Content Server Provides Content
209
Content Server Logs Request
210
Player Client Receives Content
211
Player Client Outputs Content
212
Player Client Records User Behavior and Viewing of Content
213
Player Client Transmits Transaction and User Data to Data Collection Server
214
End

FIG.2

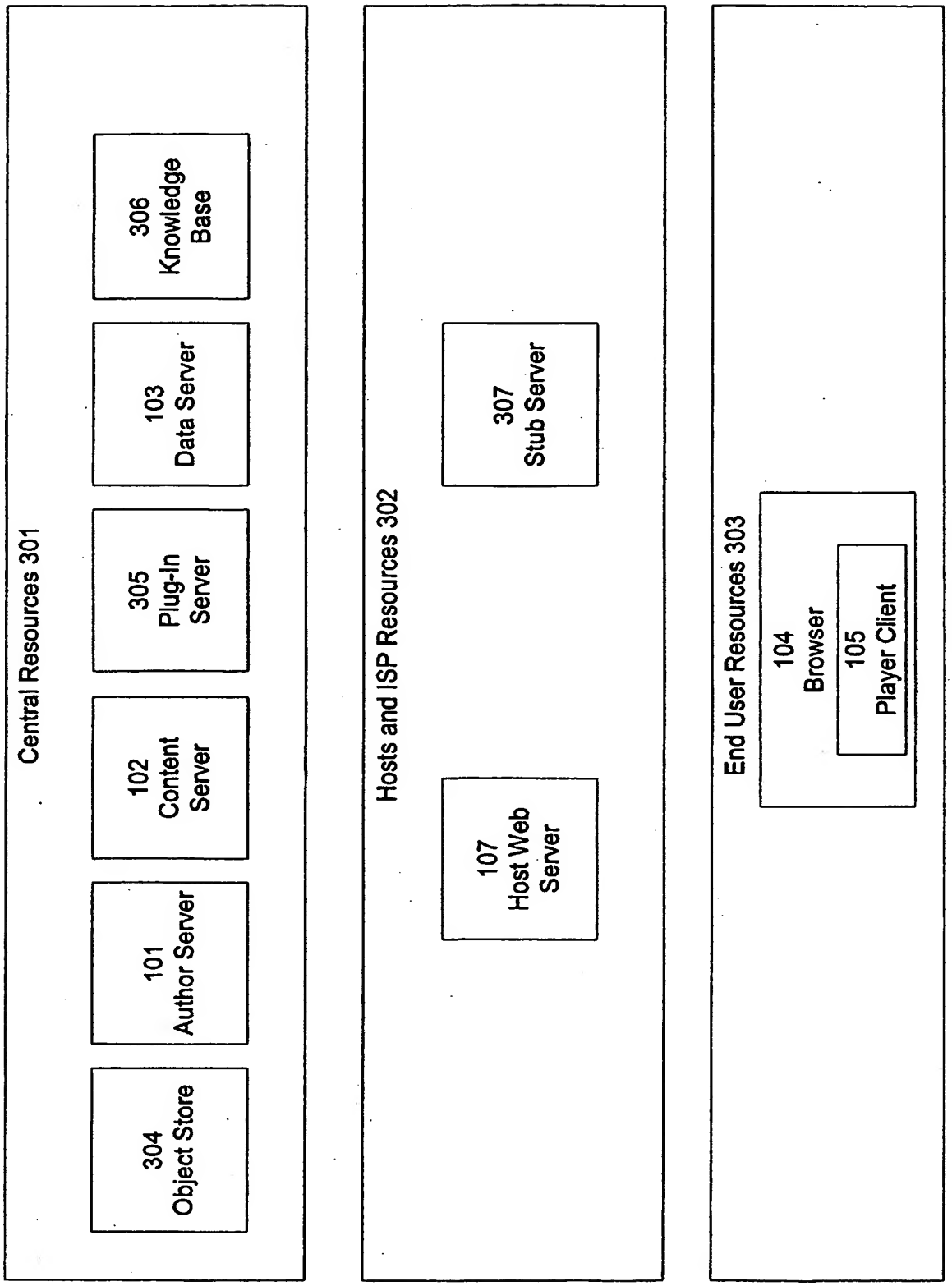

Central Resources 301
304 Object Store
101 Author Server
102 Content Server
305 Plug-In Server
103 Data Server
306 Knowledge Base
Hosts and ISP Resources 302
107 Host Web Server
307 Stub Server
End User Resources 303
104 Browser
105 Player Client

FIG. 3

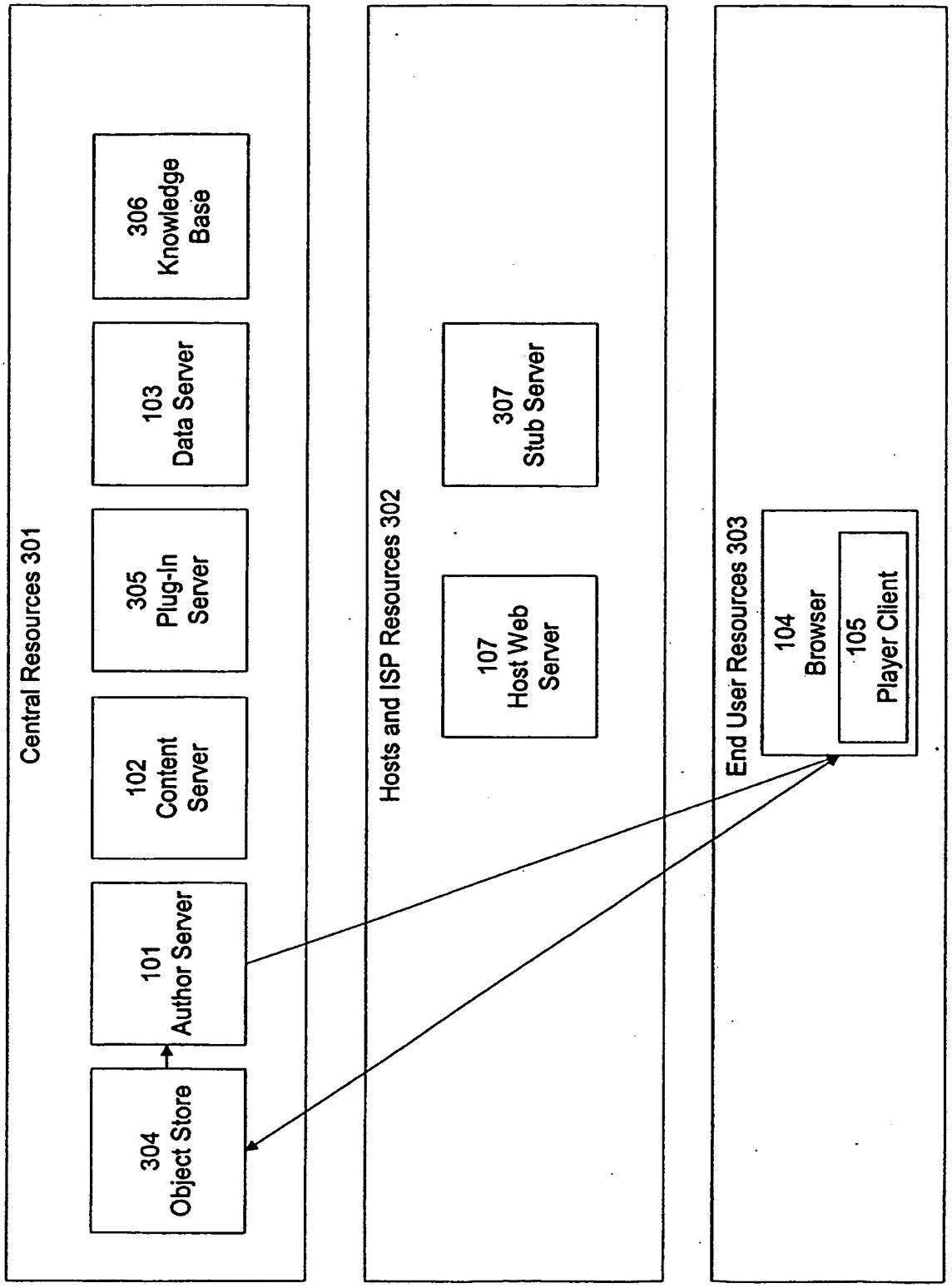

Central Resources 301
306 Knowledge Base
103 Data Server
305 Plug-In Server
102 Content Server
101 Author Server
304 Object Store
Hosts and ISP Resources 302
307 Stub Server
107 Host Web Server
End User Resources 303
104 Browser
105 Player Client

FIG. 4

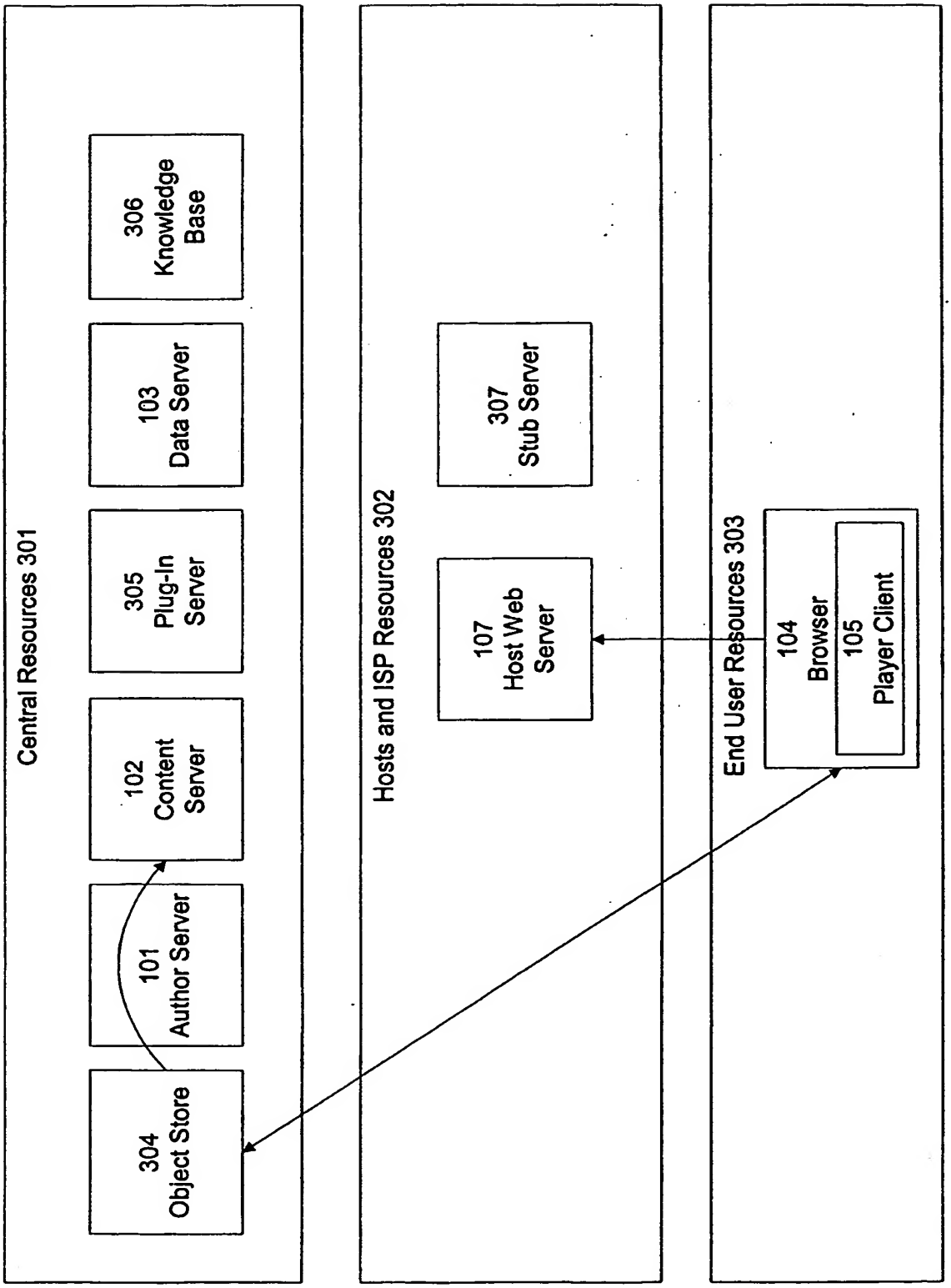

Central Resources 301
304 Object Store
101 Author Server
102 Content Server
305 Plug-In Server
103 Data Server
306 Knowledge Base
Hosts and ISP Resources 302
107 Host Web Server
307 Stub Server
End User Resources 303
104 Browser
105 Player Client

FIG. 5

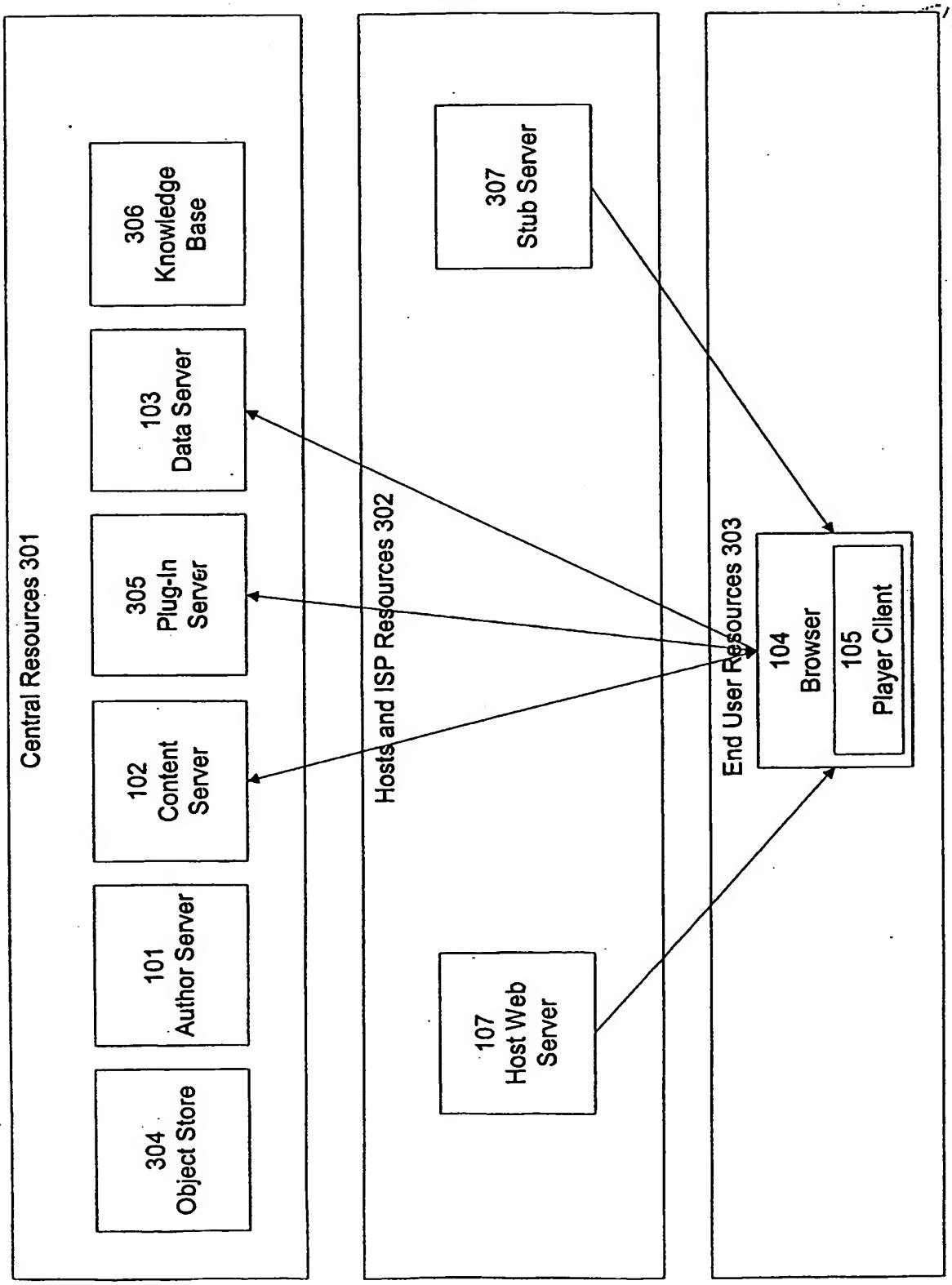

Central Resources 301
306 Knowledge Base
103 Data Server
305 Plug-In Server
102 Content Server
101 Author Server
304 Object Store
Hosts and ISP Resources 302
307 Stub Server
107 Host Web Server
End User Resources 303
104 Browser
105 Player Client

FIG. 6

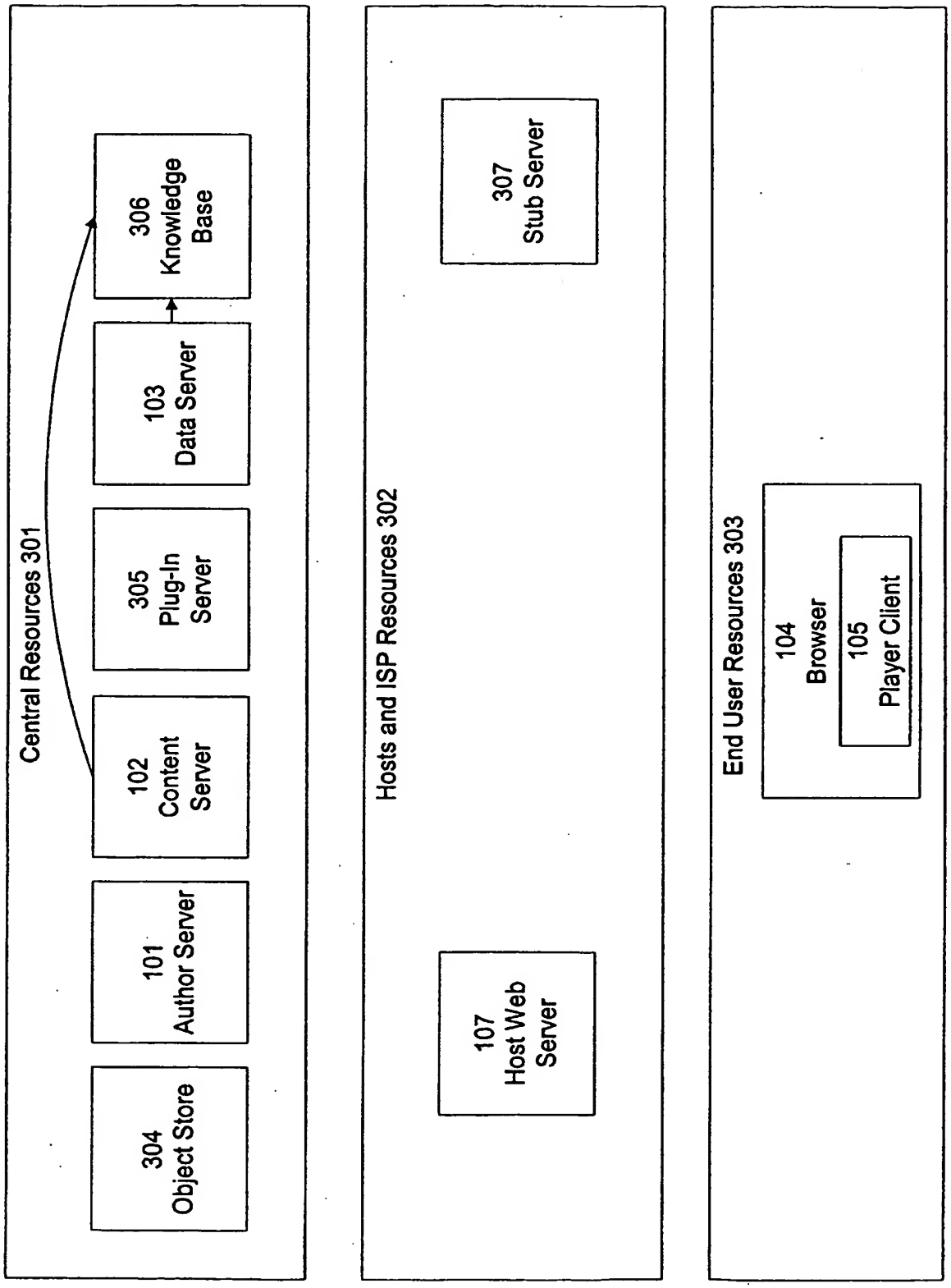

Central Resources 301
304 Object Store
101 Author Server
102 Content Server
305 Plug-In Server
103 Data Server
306 Knowledge Base
Hosts and ISP Resources 302
107 Host Web Server
307 Stub Server
End User Resources 303
104 Browser
105 Player Client

FIG. 7

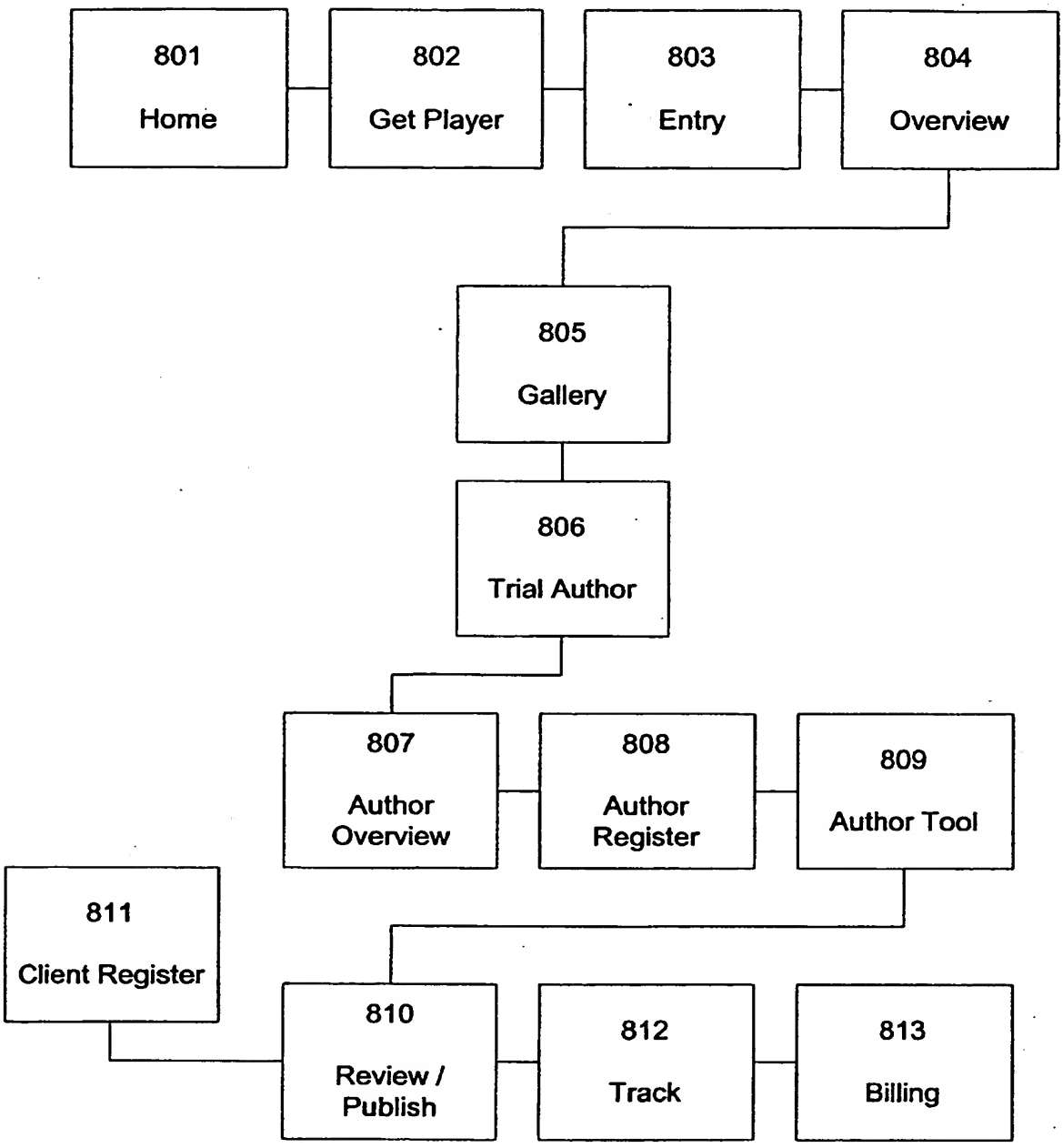
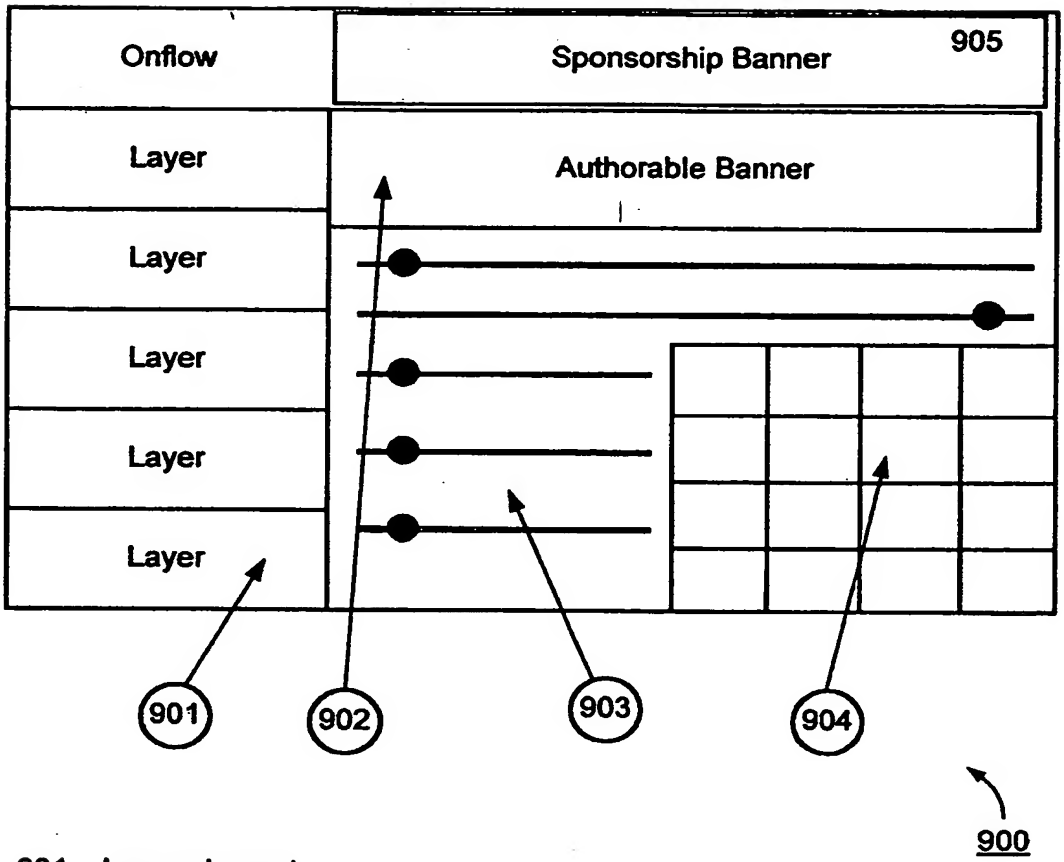

801
Home
802
Get Player
803
Entry
804
Overview
805
Gallery
806
Trial Author
807
Author Overview
808
Author Register
809
Author Tool
811
Client Register
810
Review / Publish
812
Track
813
Billing

FIG.8



901 Layer element
902 Banner being authored
903 Controls for selected layer
904 Theme sheet to modify selected layer

Figure 9

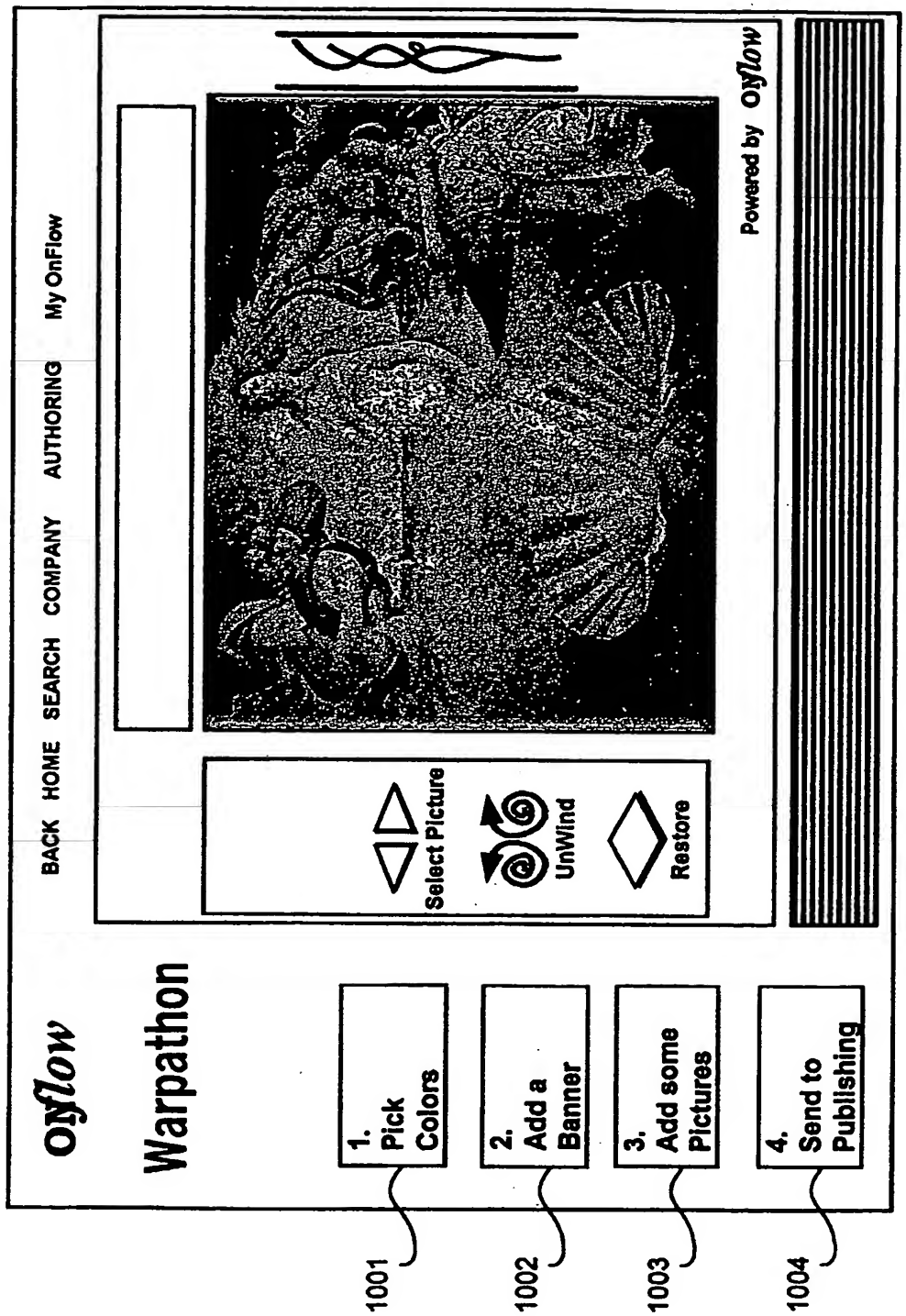

oyflow
Warpathon
BACK HOME SEARCH COMPANY AUTHORING My OnFlow
Powered by oyflow
Select Picture
UnWind
Restore
1. Pick Colors
2. Add a Banner
3. Add some Pictures
4. Send to Publishing
1001
1002
1003
1004
1000

Figure 10

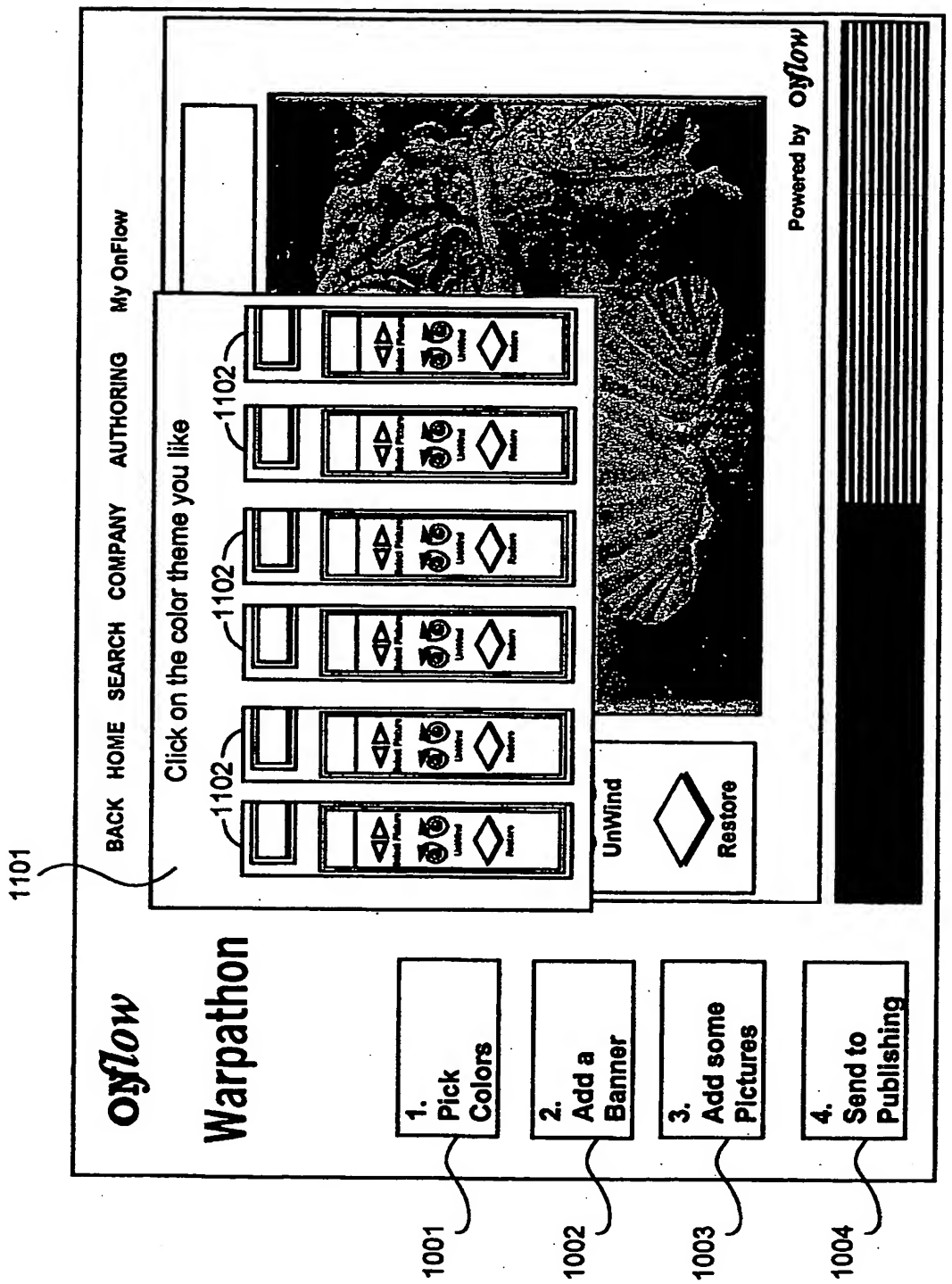

1100
1101
oyflow
Warpathon
BACK HOME SEARCH COMPANY AUTHORING My OnFlow
Click on the color theme you like
1102
1102
1102
UnWind
Restore
Powered by oyflow
1. Pick Colors
2. Add a Banner
3. Add some Pictures
4. Send to Publishing
1001
1002
1003
1004

Figure 11

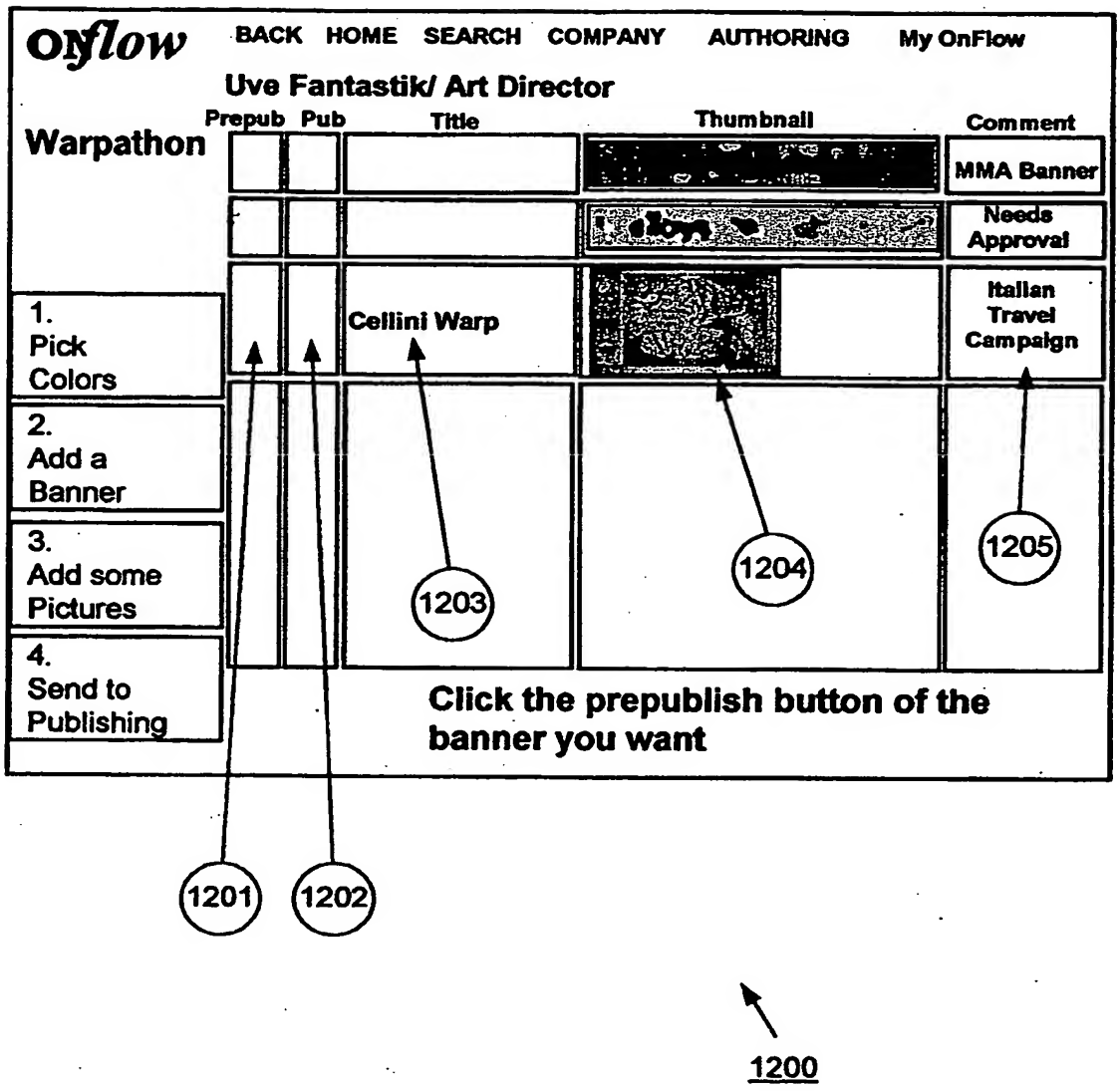

OnFlow
BACK HOME SEARCH COMPANY AUTHORING My OnFlow
Uve Fantastik/ Art Director
Warpathon
Prepub
Pub
Title
Thumbnail
Comment
MMA Banner
Needs Approval
Cellini Warp
Italian Travel Campaign
1. Pick Colors
2. Add a Banner
3. Add some Pictures
4. Send to Publishing
Click the prepublish button of the banner you want
1201
1202
1203
1204
1205
1200

Figure 12

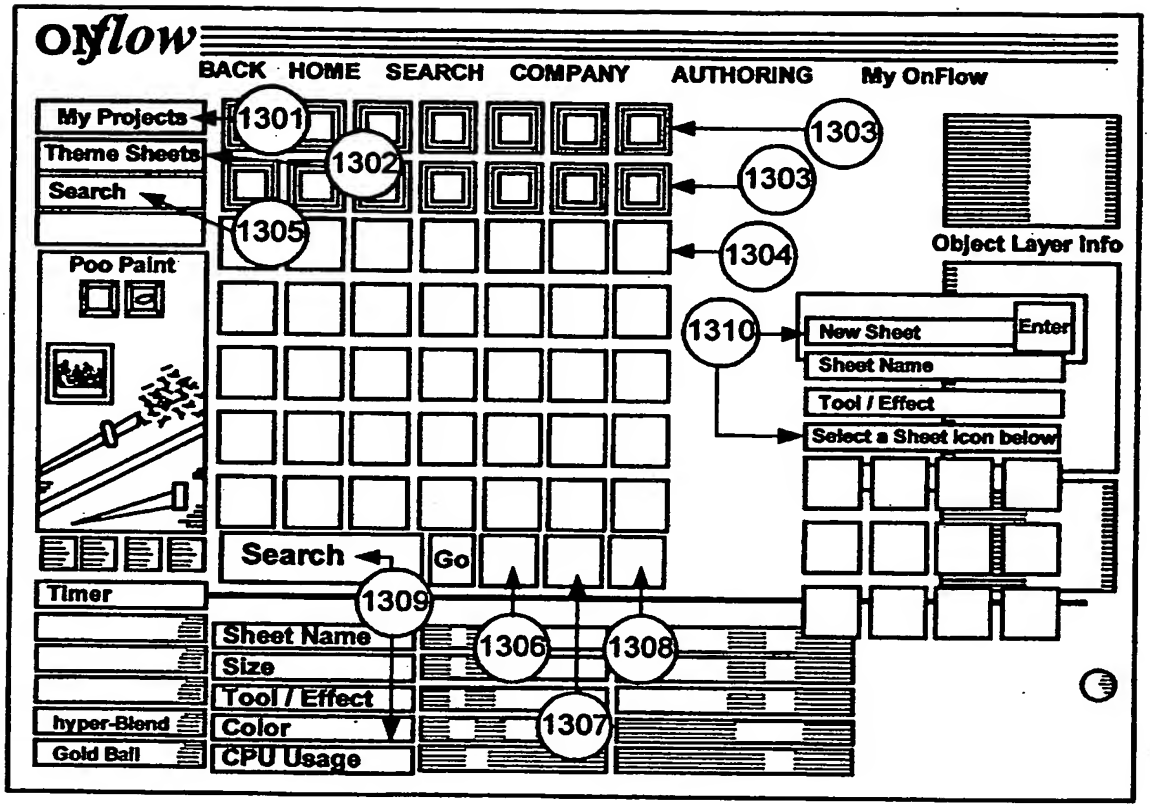

ONflow
BACK HOME SEARCH COMPANY AUTHORING My OnFlow
My Projects
Theme Sheets
Search
Poo Paint
Timer
hyper-Blend
Gold Ball
Search
Go
Sheet Name
Size
Tool / Effect
Color
CPU Usage
1301
1302
1303
1303
1304
1305
1306
1307
1308
1309
1310
Object Layer Info
New Sheet
Enter
Sheet Name
Tool / Effect
Select a Sheet icon below

Figure 13

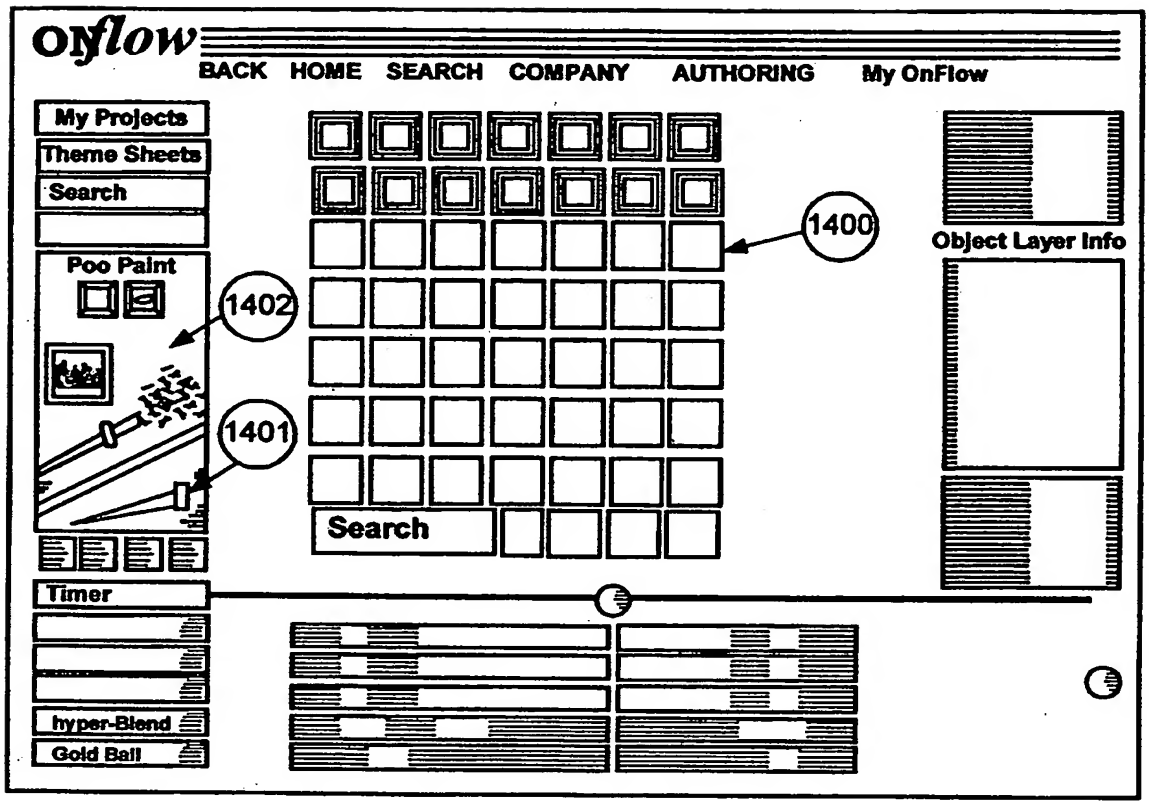

OnFlow
BACK HOME SEARCH COMPANY AUTHORING My OnFlow
My Projects
Theme Sheets
Search
Poo Paint
1402
1401
1400
Object Layer Info
Search
Timer
hyper-Blend
Gold Ball

Figure 14

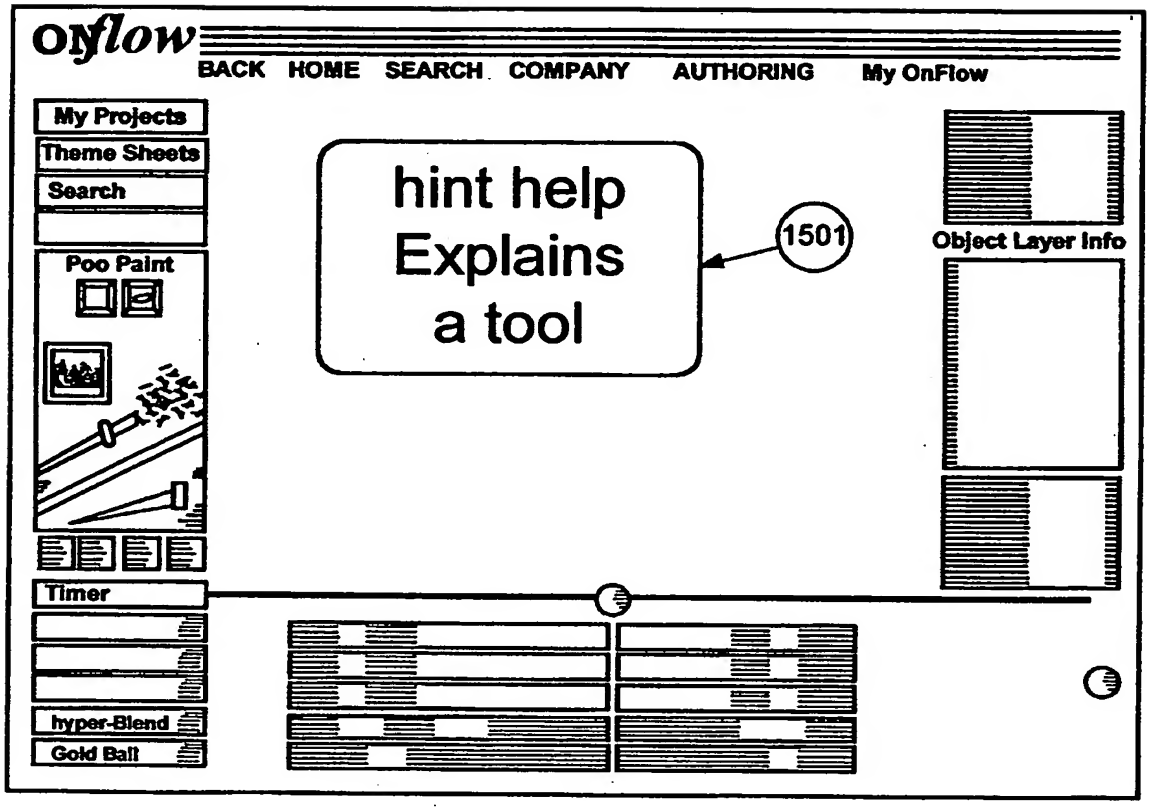

onflow
BACK HOME SEARCH COMPANY AUTHORING My OnFlow
My Projects
Theme Sheets
Search
Poo Paint
Timer
hyper-Blend
Gold Ball
hint help
Explains
a tool
1501
Object Layer Info

Figure 15

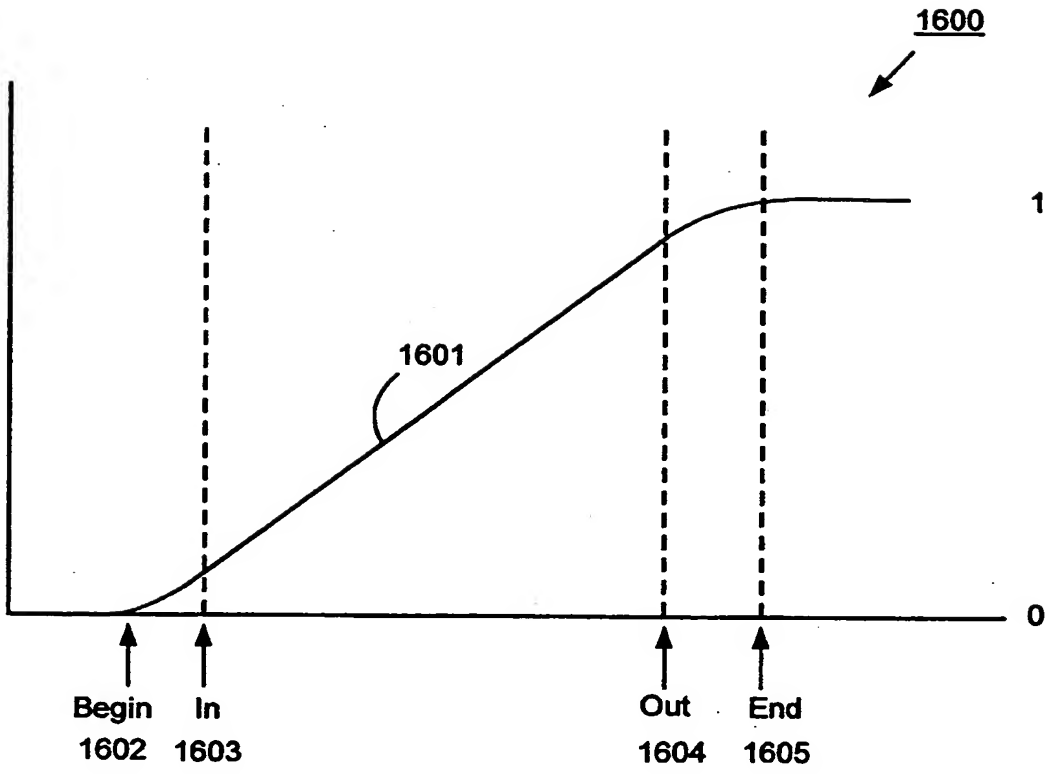


Figure 16

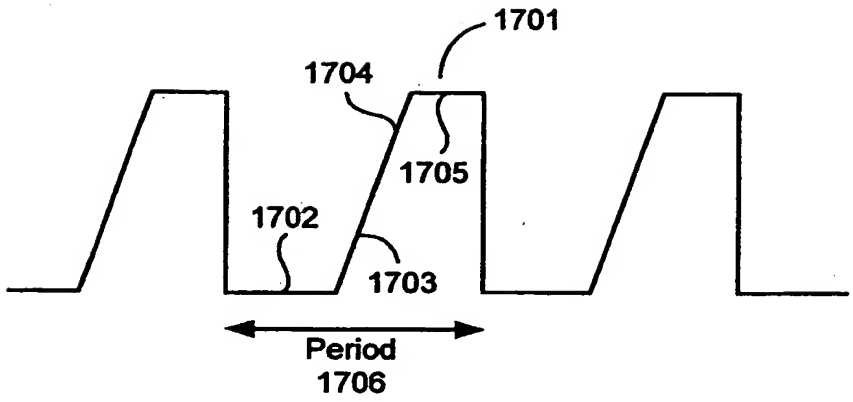


Figure 17

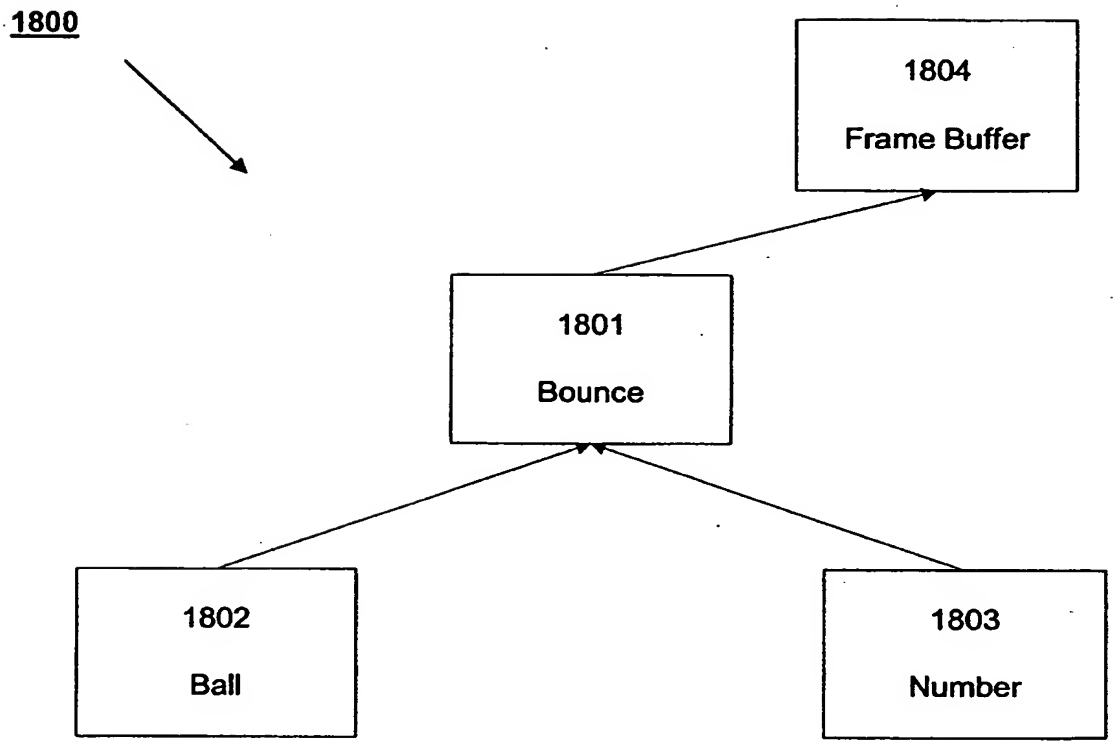


FIG. 18